67471-021
Y. YAMAMOTO
July 23, 2003.
McDermott, Will & Emery

日 本 国 特 許 庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office

出願年月日
Date of Application: ２００２年 ７月２４日

出願番号
Application Number: 特願２００２－２１５３９４

[ST.10/C]: [ＪＰ２００２－２１５３９４]

出願人
Applicant(s): 松下電器産業株式会社

２００３年 ３月１４日

特許庁長官
Commissioner,
Japan Patent Office 太田信一郎

出証番号 出証特２００３－３０１７２８２

【書類名】　特許願
【整理番号】　5037730146
【提出日】　平成14年 7月24日
【あて先】　特許庁長官　殿
【国際特許分類】　G06F　9/42
【発明者】
　【住所又は居所】　大阪府門真市大字門真１００６番地　松下電器産業株式会社内
　【氏名】　山本　泰宜
【特許出願人】
　【識別番号】　000005821
　【氏名又は名称】　松下電器産業株式会社
【代理人】
　【識別番号】　100090446
　【弁理士】
　【氏名又は名称】　中島　司朗
【手数料の表示】
　【予納台帳番号】　014823
　【納付金額】　21,000円
【提出物件の目録】
　【物件名】　明細書　1
　【物件名】　図面　1
　【物件名】　要約書　1
　【包括委任状番号】　9003742
【プルーフの要否】　要

【書類名】　　明細書

【発明の名称】　情報処理装置、情報処理方法、およびプログラム変換装置

【特許請求の範囲】

【請求項１】　演算に用いるデータを保持する１つ以上のレジスタを有し、機械語プログラムに従って処理を行うプロセッサを含む情報処理装置であって、

所定の関数の呼び出しに伴いスタックメモリへ前記レジスタの内容を圧縮して退避することを示す情報が、前記機械語プログラム中に有るか否かを識別する圧縮識別手段と、

圧縮識別手段による識別結果が肯定的である場合には所定の関数の呼び出しに伴いスタックメモリへ前記レジスタの内容を圧縮して退避し、圧縮識別手段による識別結果が否定的である場合には所定の関数の呼び出しに伴いスタックメモリへ前記レジスタの内容を圧縮せずに退避する退避手段と

を備えることを特徴とする情報処理装置。

【請求項２】　前記情報処理装置はさらに、

所定の関数の呼び出し終了に伴い前記レジスタへスタックメモリに退避されているデータを伸張して復帰することを示す情報が、前記機械語プログラム中に有るか否かを識別する伸張識別手段と、

伸張識別手段による識別結果が肯定的である場合には所定の関数の呼び出し終了に伴い前記レジスタへ前記スタックメモリに退避されているデータを伸張して復帰し、伸張識別手段による識別結果が否定的である場合には所定の関数の呼び出し終了に伴い前記レジスタへ前記スタックメモリに退避されているデータを伸張せずに復帰する復帰手段とを備えること

を特徴とする請求項１記載の情報処理装置。

【請求項３】　前記退避手段は、

所定の関数の呼び出しに伴いスタックメモリへ前記レジスタの内容を圧縮して退避する場合に、所定の関数の呼び出し終了に伴い前記レジスタへスタックメモリに退避されているデータを伸張して復帰することを示す伸張情報と、前記レジスタの内容を圧縮したデータとを対応付けてスタックメモリに退避する伸張情報退避手段を含み、

前記情報処理装置はさらに、

所定の関数の呼び出し終了に伴い前記レジスタへ復帰しようとするデータと対応付けられている伸張情報が、スタックメモリ中に有るか否かを識別する伸張識別手段と、

伸張識別手段による識別結果が肯定的である場合には所定の関数の呼び出し終了に伴い前記レジスタへスタックメモリ中の前記復帰しようとするデータを伸張して復帰し、伸張識別手段による識別結果が否定的である場合には所定の関数の呼び出し終了に伴い前記レジスタへスタックメモリ中の前記復帰しようとするデータを伸張せずに復帰する復帰手段とを備えること

を特徴とする請求項１記載の情報処理装置。

【請求項４】　前記退避手段はさらに、

前記レジスタに格納されている第１データを所定のアルゴリズムに従って、第２データに変換するデータ変換手段と、

前記第２データのデータサイズを、圧縮効率を示すしきい値と比較する比較手段と、

前記第２データのデータサイズがしきい値より小さい場合には前記第２データをスタックメモリへ退避し、前記第２データのデータサイズがしきい値より大きい場合には前記第１データをスタックメモリへ退避する選択退避手段とを含むこと

を特徴とする請求項３記載の情報処理装置。

【請求項５】　前記退避手段は、圧縮識別手段による識別結果が肯定的である場合、前記所定の関数の呼び出し元における呼び出し時にスタックメモリへ前記レジスタの内容を圧縮して退避し、

前記復帰手段は、伸張識別手段による識別結果が肯定的である場合、前記所定の関数の呼び出し元における呼び出し終了時に前記レジスタへスタックメモリに退避されているデータを伸張して復帰すること

を特徴とする請求項２、及び請求項３のいづれか１項に記載の情報処理装置。

【請求項６】　前記退避手段は、圧縮識別手段による識別結果が肯定的である場合、呼び出された前記所定の関数における処理開始時にスタックメモリへ前

記レジスタの内容を圧縮して退避し、

前記復帰手段は、伸張識別手段による識別結果が肯定的である場合、呼び出された前記所定の関数における処理終了時に前記レジスタへスタックメモリに退避されているデータを伸張して復帰すること

を特徴とする請求項２、及び請求項３のいづれか１項に記載の情報処理装置。

【請求項７】　演算に用いるデータを保持する１つ以上のレジスタを有し、機械語プログラムに従って処理を行うプロセッサを含む情報処理装置における情報処理方法であって、

所定の関数の呼び出しに伴いスタックメモリへ前記レジスタの内容を圧縮して退避することを示す情報が、前記機械語プログラム中に有るか否かを識別する圧縮識別ステップと、

圧縮識別ステップによる識別結果が肯定的である場合には所定の関数の呼び出しに伴いスタックメモリへ前記レジスタの内容を圧縮して退避し、圧縮識別ステップによる識別結果が否定的である場合には所定の関数の呼び出しに伴いスタックメモリへ前記レジスタの内容を圧縮せずに退避する退避ステップと

を備えることを特徴とする情報処理方法。

【請求項８】　プログラム変換装置であって、

１、又は複数の関数を含む入力プログラムを入手する入手手段と、

所定の関数の呼び出しに伴いスタックメモリへプロセッサ内のレジスタの内容を圧縮して退避するか、圧縮せずにそのまま退避するかを前記入力プログラムにおいて判定する判定手段と、

当該判定手段により圧縮して退避すると判定された場合、スタックメモリへ前記レジスタの内容を圧縮して退避することをプロセッサに示す情報を備える出力プログラムに、前記入力プログラムを変換する変換手段と

を備えることを特徴とするプログラム変換装置。

【請求項９】　前記判定手段は、

前記レジスタの内容を格納したスタックメモリを参照するスタックアクセス関数を検出する検出手段と、

前記スタックアクセス関数、及び前記スタックアクセス関数の呼び出し側の上

位に位置する全ての階層の関数の呼び出しに伴いスタックメモリへ前記レジスタの内容を圧縮せずにそのまま退避すると判定する除外手段とを含むこと

を特徴とする請求項８記載のプログラム変換装置。

【請求項１０】　前記判定手段は、

スタックメモリへ前記レジスタの内容を圧縮して退避することを示す情報が前記所定の関数に事前に付加されている場合、前記所定の関数の呼び出しに伴いスタックメモリへ前記レジスタの内容を圧縮して退避すると判定する事前指定判定手段を含むこと

を特徴とする請求項８記載のプログラム変換装置。

【請求項１１】　前記判定手段は、

前記所定の関数が内部にサブルーチンを持つ場合、前記入力プログラムが含む関数群の階層構造を示すネスト情報に基づいて、前記所定の関数の呼び出しに伴いスタックメモリへ前記レジスタの内容を圧縮して退避するか、圧縮せずにそのまま退避するかを判定するネスト判定手段を含むこと

を特徴とする請求項８記載のプログラム変換装置。

【請求項１２】　前記変換手段は、

前記所定の関数の呼び出しに伴いスタックメモリへ前記レジスタの内容を圧縮して退避することをプロセッサに示す情報を、前記所定の関数の呼び出し元関数に付加する圧縮情報付加手段と、

前記所定の関数の呼び出し終了に伴い前記レジスタへスタックメモリに退避されているデータを伸張し復帰することをプロセッサに示す情報を、前記所定の関数の呼び出し元関数に付加する伸張情報付加手段とを含むこと

を特徴とする請求項８記載のプログラム変換装置。

【請求項１３】　前記変換手段は、

前記所定の関数の呼び出しに伴いスタックメモリへ前記レジスタの内容を圧縮して退避することをプロセッサに示す情報を、呼び出される前記所定の関数に付加する圧縮情報付加手段と、

前記所定の関数の呼び出し終了に伴い前記レジスタへスタックメモリに退避されているデータを伸張し復帰することをプロセッサに示す情報を、呼び出される

前記所定の関数に付加する伸張情報付加手段とを含むこと

を特徴とする請求項８記載のプログラム変換装置。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は、関数の呼び出しに伴い発生するスタックメモリへのレジスタの内容の退避時にスタックメモリの使用効率を向上させる技術に関する。

【０００２】

【従来の技術】

プログラムを実行するコンピュータは、プログラムのメインルーチンから関数の呼び出しに伴い、現在実行している処理のアドレスや、レジスタなどの資源をスタックメモリに退避し、呼び出された関数からの復帰に伴い、スタックメモリからレジスタを復帰し、退避したアドレスへ処理を戻す。しかしながら、全てのレジスタを退避するとそれだけ退避に要するサイクル数やメモリ容量が大きくなりシステム性能が劣化するので、関数の前後で値を保証する必要のある保証レジスタのみを退避の対象として、退避すべきレジスタの数を削減している。

【０００３】

サイクル数削減という点では、例えばサブルーチンの呼び出し側と呼び出され側とが異なるファイルに配置されていてもパイプラインの乱れを生じることなくサブルーチンを呼び出すプログラム変換装置及びプロセッサが特開平8-305581号公報に開示されている。

【０００４】

【発明が解決しようとする課題】

しかしながら、近年のプログラムの大規模化に伴い、プロセッサが有するレジスタの本数は増大する傾向にある。従って、保証レジスタ群だけをスタックメモリへ退避するとしてもそのデータ量は大量となり、スタックメモリのオーバーフローを回避するために大容量のスタックメモリを必要とする問題がある。

【０００５】

本発明はかかる問題に鑑み、スタックメモリの使用効率を上げることでスタッ

クメモリのオーバーフローが発生する危険性を抑え、省メモリを実現する情報処理装置、情報処理方法、およびプログラム変換装置を提供することを目的とする。

【０００６】

【課題を解決するための手段】

上記目的を達成するために、本発明に係る情報処理装置は、演算に用いるデータを保持する１つ以上のレジスタを有し、機械語プログラムに従って処理を行うプロセッサを含む情報処理装置であって、所定の関数の呼び出しに伴いスタックメモリへ前記レジスタの内容を圧縮して退避することを示す情報が、前記機械語プログラム中に有るか否かを識別する圧縮識別手段と、圧縮識別手段による識別結果が肯定的である場合には所定の関数の呼び出しに伴いスタックメモリへ前記レジスタの内容を圧縮して退避し、圧縮識別手段による識別結果が否定的である場合には所定の関数の呼び出しに伴いスタックメモリへ前記レジスタの内容を圧縮せずに退避する退避手段とを備えることを特徴とする。

【０００７】

また、本発明に係る情報処理方法は、演算に用いるデータを保持する１つ以上のレジスタを有し、機械語プログラムに従って処理を行うプロセッサを含む情報処理装置における情報処理方法であって、所定の関数の呼び出しに伴いスタックメモリへ前記レジスタの内容を圧縮して退避することを示す情報が、前記機械語プログラム中に有るか否かを識別する圧縮識別ステップと、圧縮識別ステップによる識別結果が肯定的である場合には所定の関数の呼び出しに伴いスタックメモリへ前記レジスタの内容を圧縮して退避し、圧縮識別ステップによる識別結果が否定的である場合には所定の関数の呼び出しに伴いスタックメモリへ前記レジスタの内容を圧縮せずに退避する退避ステップとを備えることを特徴とする。

【０００８】

これによって、関数の呼び出しに伴いスタックメモリへ退避するレジスタの内容を圧縮することができるので、関数の呼び出しの前後でのレジスタの値を保証したうえで、スタックメモリに格納するデータ量を削減することができる。

従って、スタックメモリの使用効率を上げることでスタックメモリのオーバー

フローが発生する危険性を抑え、省メモリを実現することができる。

【０００９】

上記目的を達成するために、本発明に係わるプログラム変換装置は、プログラム変換装置であって、１、又は複数の関数を含む入力プログラムを入手する入手手段と、所定の関数の呼び出しに伴いスタックメモリへプロセッサ内のレジスタの内容を圧縮して退避するか、圧縮せずにそのまま退避するかを前記入力プログラムにおいて判定する判定手段と、当該判定手段により圧縮して退避すると判定された場合、スタックメモリへ前記レジスタの内容を圧縮して退避することをプロセッサに示す情報を備える出力プログラムに、前記入力プログラムを変換する変換手段とを備えることを特徴とする。

【００１０】

これによって、呼び出しに伴いスタックメモリへ退避するレジスタの内容を圧縮するのに適した関数を、圧縮対象とすることができる。

従って、スタックメモリの使用効率を上げることができるプログラムを実現することができる。

【００１１】

【発明の実施の形態】

以下、本発明の実施の形態について、図１から図２６を用いて説明する。

＜概要＞

本発明は、関数の呼び出しに伴いスタックメモリに退避する保証レジスタの内容を圧縮することで、スタックメモリの効率的な使用を実現するものである。

【００１２】

以下では、まず、プログラムに従って動作するコンピュータであり、関数の呼び出し、復帰に伴いスタックへ退避、復帰する保証レジスタの内容を圧縮、伸張するかを示す情報を付加された実行ファイル２０を生成するプログラム変換装置１００と関連ファイル群について説明し、続いて、プログラムに従って動作するコンピュータであり、プログラム変換装置１００が生成する実行ファイル２０を実行する情報処理装置２００について説明する。

【００１３】

＜構成＞

図１は、本発明の実施の形態に係わるプログラム変換装置、及び情報処理装置を備えた情報処理システムの構成を示す図である。

図に示す情報処理システムは、ソースファイル群１０を変換し、実行ファイル２０を生成するプログラム変換装置１００と、実行ファイル２０を実行する情報処理装置２００とから構成される。

【００１４】

（ソースファイル）

ソースファイル群１０は、ここではソースファイル１１、及びソースファイル１２の２つのソースファイルから構成される。

図２は、ソースファイル１１、図３はソースファイル１２の一例を模式的に示す図である。

【００１５】

ソースファイル１１、ソースファイル１２は高級言語により記述されたプログラムファイルであり、ソースファイル１１は関数mainに関する記述部１１ａと、関数func#aに関する記述部１１ｂと、関数func#bに関する記述部１１ｃと、関数func#dに関する記述部１１ｄと、関数func#eに関する記述部１１ｅと、関数func#fに関する記述部１１ｇとを含み、ソースファイル１２は関数func#cに関する記述部１２ａを含む。ここで記述される関数群は呼び出し、及び復帰により処理を分岐するツリー構造を持つ。

【００１６】

関数mainに関する記述部１１ａは、関数func#aを呼び出す命令と、関数func#bを呼び出す命令と、関数func#eを呼び出す命令とを含む。関数func#aに関する記述部１１ｂは関数func#cを呼び出す命令を含む。関数func#cに関する記述部１２ａは関数func#fを呼び出す命令を含む。関数func#bに関する記述部１１ｃは関数func#cを呼び出す命令と、関数func#dを呼び出す命令と、前記２つの命令のどちらか一方に条件分岐する判定とを含む。関数func#dに関する記述部１１ｄは、インラインアセンブラ記述された、スタックポインタが指し示すアドレスから４バイト上位のアドレスのメモリ内容をレジスタｒ０に転送するロード命令と、レジ

スタｒ０の内容と“0xf0”の論理積をレジスタｒ０に格納する論理積演算命令と、レジスタｒ０の内容をスタックポインタから４バイト上位のアドレスに転送するストア命令を含む。関数func#eに関する記述部１１ｅは、プラグマ#STACK#COMPRESSの記述部１１ｆを含む。ここでプラグマ#STACK#COMPRESSとは関数の呼び出し時に保証レジスタからスタックメモリへ退避する退避データを圧縮することを、プログラム変換装置１００に明示するプラグマである。

【００１７】

（プログラム変換装置）

図４は、本発明の実施の形態に係わるプログラム変換装置の構成を示す図である。

プログラム変換装置１００は、コンパイル部１１０、中間コード保持部１２０、選択候補情報保持部１３０、リンク部１４０、及び記憶部１５０から構成される。

【００１８】

コンパイル部１１０はソースファイル群１０を取得し、中間コード、及び選択候補テーブル１３１を作成するコンパイラであり、ソースファイル取得部１１１、翻訳部１１２、スタックアクセス検出部１１３、及びユーザ指定関数抽出部１１４から構成される。

ソースファイル取得部１１１は、記憶部１５０が保持するソースファイル１１、及びソースファイル１２を取得し、翻訳部１１２、スタックアクセス検出部１１３、及びユーザ指定関数抽出部１１４にソースファイル１１、及びソースファイル１２を渡す。

【００１９】

翻訳部１１２は、ソースファイル１１、及びソースファイル１２に記述される各関数を、呼び出し元の関数から処理を分岐するコール命令、関数内の処理を行う機械語命令、及び呼び出し元の関数へ処理を復帰するリターン命令に変換して、ソースファイル１１、及びソースファイル１２から機械語の中間コード群を生成する。翻訳部１１２は、生成した中間コード群を中間コード保持部１２０へ出力する。

【００２０】

スタックアクセス検出部１１３は、ソースファイル１１、及びソースファイル１２に含まれる各関数において各関数内の処理を調査し、各関数の処理開始時のスタックポインタの値より上位のアドレスを参照する記述がある関数を検出する。スタックアクセス検出部１１３は検出した関数の名前と関数評価値を選択候補情報保持部１３０が保持する選択候補テーブル１３１に書き込む。ここで検出された関数の関数評価値は０とされる。

【００２１】

ユーザ指定関数抽出部１１４は、ソースファイル１１、及びソースファイル１２に含まれる関数のうち、スタックアクセス検出部１１３によって選択候補テーブル１３１に書き込まれていない関数においてプラグマ#STACK#COMPRESSの記述の有無を調査する。ユーザ指定関数抽出部１１４は、ここで調査した関数のうちプラグマ#STACK#COMPRESSが記述された関数の名前と関数評価値を２として選択候補テーブル１３１に書き込み、また、プラグマ#STACK#COMPRESSが記述されていない関数の名前と関数評価値を１として選択候補テーブル１３１に書き込む。

【００２２】

中間コード保持部１２０は機械語の中間コード群を保持するメモリ領域である。

選択候補情報保持部１３０は選択候補テーブル１３１を保持するメモリ領域である。

図５は、選択候補情報保持部１３０が保持する選択候補テーブル１３１のデータ構造を示す図である。

【００２３】

選択候補テーブル１３１はソースファイル群１０に含まれる関数の数と同数の選択候補情報から構成される。各選択候補情報は、各関数と１対１に対応し、関数名１３１ａ、及び関数評価値１３１ｂから構成される。関数評価値１３１ｂは、０、１、及び２の何れかの値をとり、スタックアクセス検出部１１３で検出された、関数の処理開始時のスタックポインタが示すアドレスより、上位のアドレスのスタックメモリにアクセスする処理がある関数、つまり、既にいずれかの関

数の呼び出しによって保証レジスタの値が格納されているスタックメモリを参照する関数の関数評価値１３１ｂは０、ユーザ指定関数抽出部１１４で検出された、プラグマ#STACK#COMPRESSが記述された関数の関数評価値１３１ｂは２、上記２種類以外の関数の関数評価値１３１ｂは１となる。

【００２４】

リンク部１４０は中間コード、及び選択候補テーブル１３１から実行ファイル２０を作成するリンカであり、連結部１４１、関数ツリー情報作成部１４２、関数ツリー情報保持部１４３、関数選択部１４４、及び圧縮・伸張ビット付加部１４５から構成される。

連結部１４１は中間コード保持部１２０に保持される中間コード群に実行時の再配置情報などを追加しリンクすることで、実行可能な機械語コードを生成し圧縮・伸張ビット付加部に出力する。

【００２５】

図６は、呼び出し関係にある関数のツリー構造を示す図である。各関数から見て、図の横軸の左方向にある呼び出し元の関数をツリー構造の上位の関数、右方向にある呼び出し先の関数をツリー構造の下位の関数と呼ぶ。また、図に示すようにメインルーチンである関数mainからは、関数func#a、関数func#b、及び関数func#eの３つのサブルーチンが呼び出され、これら３つのサブルーチンの先頭関数をツリー構造の最上位の関数とする。

【００２６】

関数ツリー情報作成部１４２は中間コード保持部１２０に保持される中間コード群から、自らより下位の関数を持たない最下位の関数を検出し、最下位の関数から見て最上位の関数までの呼び出し関係をツリー構造として抽出する。ここでは、関数func#a、関数func#c、及び関数func#fが含まれるツリー構造と、関数func#b、関数func#c、及び関数func#fが含まれるツリー構造と、関数func#b、及び関数func#dが含まれるツリー構造と、関数func#eのみのツリー構造とが抽出される。関数ツリー情報作成部１４２は抽出した各ツリー構造についてツリー番号を付し、ツリー構造を構成する各関数の関数評価値１３１ｂの積算値であるツリー評価値を算出する。以上の情報をもとに関数ツリー情報作成部１４２はツリー番

号とツリー評価値を関数ツリー情報保持部１４３に書き込み、関数ツリー情報テーブル１４３ａを作成する。

【００２７】

図７は、関数ツリー情報保持部１４３で保持する関数ツリー情報テーブル１４３ａのデータ構造を示す図である。

関数ツリー情報テーブル１４３ａは中間コード中の関数のツリー構造の数と同数のツリー情報から構成される。各ツリー情報は各ツリー構造と１対１で対応し、ツリー番号１４３ｂ、及びツリー評価値１４３ｃから構成される。

【００２８】

ツリー評価値１４３ｃはツリー構造を構成する各関数の関数評価値１３１ｂを互いに乗じた値である。

ここでは、ソースファイル１１、及びソースファイル１２に基づいて作成された関数ツリー情報テーブル１４３ａに、ツリー番号１４３ｂが１、２、３、及び４である４つのツリー情報が含まれる。

【００２９】

関数選択部１４４は、ツリー評価値１４３ｃが０ではないツリー構造、つまりツリー構造を構成する関数の中に、すでにいずれかの関数の呼び出しによって保証レジスタの値が格納されているスタックメモリを参照する関数を含まないツリー構造の最上位の関数、及び関数評価値１３１ｂが２の関数、つまりユーザ指定を受けた関数を圧縮対象として選択する。

【００３０】

圧縮・伸張ビット付加部１４５は、連結部１４１で中間ファイル群をリンクし生成した機械語コードにおいて、関数選択部１４４で選択した関数のコール命令に保証レジスタの圧縮を示す１ビットの圧縮制御ビット、及びリターン命令にスタックの伸張を示す１ビットの伸張制御ビットを付加し実行ファイル２０として記憶部１５０に出力する。

【００３１】

図８は、実行ファイル２０の内容を模式的に示す図である。

実行ファイル２０では、関数func#aへ分岐するコール命令３０１と、関数func

#eへ分岐するコール命令３０２に圧縮制御ビットが付加され、関数func#aから復帰するリターン命令３０３と、関数func#eから復帰するリターン命令３０４に伸張制御ビットが付加されている。

【００３２】

ここで、コール命令は“CALL　label,Regs”のようにニーモニックで表され、保証レジスタRegsの内容をスタックメモリに退避し、labelに処理を分岐する。１ビットの圧縮制御ビットが付加されたコール命令“CALL　label,Regs,stack#compress”は保証レジスタRegsの内容を圧縮してスタックメモリに退避し、処理をlabelに分岐する。

【００３３】

リターン命令は“RET　Regs”のようにニーモニックで表され、スタックメモリの内容を保証レジスタRegsに復帰し、スタックメモリに格納されたアドレスに処理を分岐する。１ビットの伸張制御ビットが付加されたリターン命令“RET　Regs,stack#compress”はスタックメモリの内容を伸張して保証レジスタRegsに復帰し、処理をスタックメモリに格納されたアドレスに分岐する。

【００３４】

尚、本実施の形態においては、コール命令、及びリターン命令において退避、復帰すべき保証レジスタを明示したが、関数への分岐命令であるコール命令、及びリターン命令において保証レジスタの退避、復帰を行わず、ストア命令、及びロード命令によって、分岐する前にレジスタを退避、復帰することも可能である。また、分岐後にコーリー側でレジスタの退避、復帰を行うことも可能である。これらの場合、関数の呼び出しと復帰に伴う保証レジスタの退避、復帰時に保証レジスタの内容を圧縮、伸張することを示す１ビットの情報“stack#compress”は、ストア命令、及びロード命令に付加される。

【００３５】

図９は、分岐後にコーリー側で保証レジスタの退避、復帰を行う実行ファイルの内容を模式的に示す図である。

図９では、関数func#aの処理開始時に保証レジスタｒ０～ｒ７の内容をスタックポインタが示すアドレスに転送するストア命令３０５に圧縮制御ビットが付加

され、関数func#aの処理終了時にスタックポインタが指すメモリの内容を保証レジスタｒ０～ｒ７に転送するロード命令３０６に伸張制御ビットが付加される。

【００３６】

尚、本実施の形態で退避するレジスタは、保証レジスタであるｒ０～ｒ7の８個のレジスタであるが、保証レジスタとされるレジスタは８個とは限らず、それ以外の数のレジスタを保証レジスタとしてもよい。

図４に示す記憶部１５０は、メモリであり、ソースファイル群１０、及び上述の構成により生成される実行ファイル２０を保持する。

【００３７】

尚、本実施の形態におけるプログラム変換装置１００は、翻訳部１１２で変換した機械語の中間コード群を中間コード保持部１２０で保持するが、機械語の中間コード群を中間ファイルとして記憶部１５０や、外部の記録媒体に出力する構成としてもよい。

また、本実施の形態におけるプログラム変換装置１００は、ソースファイル群１０を記憶部１５０に保持しているが、ソースファイル群１０を外部の記録媒体から取得する構成としてもよい。

また、本実施の形態におけるプログラム変換装置１００は、実行ファイル２０を記憶部１５０に出力するが、実行ファイル２０を外部の記録媒体へ出力する構成としてもよい。

【００３８】

（情報処理装置）

図１０は、本発明の実施の形態に係わる情報処理装置の構成を示す図である。

情報処理装置２００は、命令メモリ２１０、処理部２２０、データメモリ２３０、圧縮部２４０、ＣＳＲ２５０、及び伸張部２６０から構成される。

命令メモリ２１０はプログラム変換装置１００により生成された圧縮・伸張制御ビット付きの実行ファイル２０を格納している。

【００３９】

処理部２２０は実行ファイル２０を実行する中央演算装置である。

図１１は、処理部２２０の内部の構成を示す図である。

処理部２２０は、ＩＦ２２１、ＤＥＣ２２２、制御部２２３、実行部２２４、レジスタ２２５、ＰＣ２２６、及びＳＰ２２７から構成される。

ＩＦ２２１は命令フェッチ部であり、命令メモリ２１０に格納されている実行ファイル２０から機械語命令をフェッチし、ＤＥＣ２２２へ渡す。

【００４０】

ＤＥＣ２２２は命令デコーダであり、機械語命令を解析する。ＤＥＣ２２２は内部に圧縮制御ビット付きコール命令デコード部２２２ａ、伸張制御ビット付きリターン命令デコード部２２２ｂを備え、圧縮制御ビット付きコール命令をデコードした場合は、制御部２２３にレジスタ２２５が保持するデータの圧縮の制御を指示し、伸張制御ビット付きリターン命令をデコードした場合は、制御部２２３にスタックメモリ２３１が保持するデータの伸張の制御を指示する。

【００４１】

尚、本実施の形態においては、コール命令、リターン命令に圧縮・伸張制御ビットが付加されている構成であるが、ストア命令、ロード命令に圧縮・伸張制御ビットが付加されている構成でもよい。この場合、ＤＥＣ２２２は内部に圧縮制御ビット付きコール命令デコード部２２２ａに替えて圧縮制御ビット付きストア命令デコード部２２２ｃを、伸張制御ビット付きリターン命令デコード部２２２ｂに替えて伸張制御ビット付きロード命令デコード部２２２ｄを備える。

【００４２】

図１２は、圧縮・伸張制御ビットが付加されているストア命令、ロード命令をデコードするＤＥＣ２２２の内部の構成を示す図である。

図１１に示す制御部２２３はＤＥＣ２２２で解析された機械語命令に従い、実行部２２４、圧縮部２４０、伸張部２６０を制御する。

コール命令時に保証レジスタ２２５ａのデータを圧縮してスタックメモリ２３１へ退避する場合は、先ずＰＣ２２６が保持するデータを、次に圧縮部２４０が保証レジスタ２２５ａのデータを圧縮したデータを、最後にＣＳＲ２５０が保持するデータをスタックメモリ２３１へ格納し、コール先の関数へ処理を分岐するように制御する。

【００４３】

リターン命令時にスタックメモリ２３１のデータを伸張して保証レジスタ２２５ａへ復帰する場合は、先ずＣＳＲ２５０のデータを、次に伸張部２６０にＣＳＲ２５０が示すサイズの保証レジスタ２２５ａのデータを、最後にＰＣ２２６のデータをスタックメモリ２３１から復帰させ、ＰＣ２２６が示す呼び出し元の関数へ処理を分岐するように制御する。

【００４４】

実行部２２４は制御部２２３の制御により機械語命令を実行する演算器である。

レジスタ２２５は３２ビットの記録装置ｒ０～ｒ３１の３２個からなり、実行部２２４が演算対象とするデータを保持する。レジスタ２２５の一部、ｒ０～ｒ７の８個からなる合計２５６ビットは保証レジスタ２２５ａであり、コール命令時にデータをスタックメモリ２３１へ退避し、リターン命令時にスタックメモリ２３１からデータを復帰することで関数の呼び出しの前後で値を保証する。

【００４５】

ＰＣ２２６はプログラムカウンタであり、現在実行している機械語命令が命令メモリ２１０上のどのアドレスかを示す。

ＳＰ２２７はスタックポインタであり、スタックメモリ２３１の未使用領域の先頭アドレスを保持する。

図１０に示すデータメモリ２３０はデータを保持するメモリである。

【００４６】

スタックメモリ２３１は、関数や割込み時の戻りアドレスや、保証レジスタの値の格納に使われるデータメモリ２３０上の領域である。図１３は、スタックメモリ２３１を模式的に示す図である。

ここでは、“0x??”と表記する事で“??”の値が１６進数であり、“0b??”と表記する事で“??”の値が２進数でありであることを表すこととする。

【００４７】

縦軸に付された値（“0xffff～0x0000”）はスタックメモリ２３１上のアドレスである。スタックメモリ２３１にデータが格納される場合、データはスタックメモリ２３１の最上位のアドレス“0xffff”から順次アドレスの下位の方向へ格

納され、スタックメモリ２３１からデータが取り出される場合、データはスタックメモリ２３１の未使用領域の先頭アドレスの１つ前のアドレスから、順次アドレスの上位の方向へ取り出される。また、スタックメモリ２３１へのデータの格納時には格納されるデータのサイズ分だけＳＰ２２７は保持する値を減らし、スタックメモリ２３１からのデータの取り出し時には取り出されるデータのサイズ分だけＳＰ２２７は保持する値を増やす。これにより、ＳＰ２２７は常にスタックメモリ２３１の未使用領域の先頭アドレスを指し示す。

【００４８】

本実施の形態においては、スタックメモリ２３１は関数の引数、及びＰＣ２２６、保証レジスタ２２５ａ、ＣＳＲ２５０等のデータを格納し、FILO（先入れ後出し）の動作により関数の呼び出し、復帰に伴う処理の分岐を実現する。

図１４は、保証レジスタ２２５ａのデータを退避したスタックメモリ２３１を模式的に示す図である。

【００４９】

圧縮部２４０は制御部２２３の制御により、スタックメモリ２３１へ退避する保証レジスタ２２５ａのデータの圧縮を行う。

図１５は、圧縮部２４０の内部の構成を示す図である。

圧縮部２４０はデータ圧縮部２４１、及び圧縮制御部２４２から構成される。

データ圧縮部２４１は保証レジスタ２２５ａのデータをハフマン符号化する。

【００５０】

図１６は、ハフマン符号対応表を示す図である。

ハフマン符号は出現率の高い値にビット長の短い符号を割り当て、出現率の低い値にビット長の長い符号を割り当てることで、データ圧縮を行う。

図１７は、図１４と同じ保証レジスタ２２５ａのデータを、図１６のハフマン符号対応表を基に圧縮して退避したスタックメモリ２３１を模式的に示す図である。

【００５１】

図１５に示す圧縮制御部２４２は内部に圧縮比較部２４３、しきい値保持部２４４を含み、圧縮比較部２４３において、しきい値保持部２４４が保持するしき

い値とデータ圧縮部２４１がハフマン符号化したデータのデータサイズを比較する。比較の結果、データ圧縮部２４１がハフマン符号化したデータのデータサイズがしきい値より小さい場合、ハフマン符号化したデータを退避データとしてスタックメモリ２３１へ格納し、ＣＳＲ２５０においてイネーブルビットを１に設定し退避データのデータサイズを格納する。比較の結果、データ圧縮部２４１がハフマン符号化したデータのデータサイズがしきい値より大きい場合、保証レジスタ２２５ａのデータを退避データとしてスタックメモリ２３１へ格納し、ＣＳＲ２５０においてイネーブルビットを０に設定する。

【００５２】

図１０に示すＣＳＲ２５０は３２ビットの記録装置であり、１ビットのイネーブルビットと、３１ビットのデータフィールドから構成される。

図１８は、ＣＳＲ２５０のビット割付けを示す図である。

横軸に付された値はＣＳＲ上のビット位置を表す。ビット［０］はスタックメモリ２３１に退避するデータが圧縮されたデータであるか否かを示すイネーブルビットであり、圧縮されたデータの場合は１、圧縮されたデータではない場合は０の値をとる。ビット［１］～［３１］は圧縮されたデータのデータサイズを格納するデータフィールドであり、イネーブルビットが１の場合は圧縮されたデータのデータサイズを、イネーブルビットが０の場合は不定値を格納する。

【００５３】

伸張部２６０は制御部２２３の制御により、保証レジスタ２２５ａへ復帰するスタックメモリ２３１のデータの伸張を行う。

図１９は、伸張部２６０の内部の構成を示す図である。

伸張部２６０は伸張制御部２６１、及びデータ伸張部２６２から構成される。

伸張制御部２６１は、ＣＳＲ２５０のイネーブルビットが１の場合は、スタックメモリ２３１に格納されている退避データをデータ伸張部２６２で伸張させて保証レジスタ２２５ａに格納し、ＣＳＲ２５０のイネーブルビットが０の場合は、スタックメモリ２３１に格納されている退避データをそのまま保証レジスタ２２５ａに格納する。

【００５４】

データ伸張部２６２は、スタックメモリ２３１に格納されているハフマン符号化データにハフマン符号化の逆変換を施しデータを伸張する。

図２０は、保証レジスタ２２５ａとスタックメモリ２３１間のデータ退避、復帰に伴うデータの流れを示す図である。

通常のコール命令時、保証レジスタ２２５ａのデータはスタックメモリ２３１へ退避され（データフローＤ１０１）、リターン命令時に、スタックメモリ２３１から保証レジスタ２２５ａへデータが復帰される（データフローＤ１０２）。

【００５５】

圧縮制御ビット付きコール命令時、保証レジスタ２２５ａのデータは圧縮部２４０で圧縮される（データフローＤ１０３）、圧縮したデータのデータサイズがしきい値保持部２４４で保持されるしきい値より大きければ、保証レジスタ２２５ａのデータがそのまま退避データとして、圧縮したデータのデータサイズがしきい値より小さければ、圧縮したデータが退避データとしてスタックメモリ２３１へ格納される（データフローＤ１０４）。このとき、圧縮結果、圧縮データのサイズがＣＳＲ２５０へ設定され（データフローＤ１０５）、ＣＳＲ２５０のデータがスタックメモリ２３１へ格納される（データフローＤ１０６）。

【００５６】

伸張制御ビット付きリターン命令時は、スタックメモリ２３１からＣＳＲ２５０へデータが復帰され（データフローＤ１０６）、ＣＳＲ２５０のデータからスタックメモリ２３１に格納されている退避データが圧縮されているか否かの情報と、退避データのデータサイズが伸張部２６０に取得され（データフローＤ１０７）、ＣＳＲ２５０が示すサイズの退避データが伸張部２６０に取得される（データフローＤ１０８）。退避データが圧縮されている場合は退避データを伸張部２６０で伸張したデータが、圧縮されていない場合は退避データがそのまま保証レジスタ２２５ａへ復帰される（データフローＤ１０９）。

【００５７】

尚、本実施の形態においては、退避データの変換アルゴリズムとして、ハフマン符号を用いているが、ラン・レングス手法等、他の可逆符号方式の圧縮アルゴリズムを用いて退避データを変換してもよい。

＜動作＞

（プログラム変換装置）

ここで、上述のように構成されたプログラム変換装置１００の動作について説明する。

【００５８】

図２１は、プログラム変換装置１００の動作の流れを示す図である。

先ず、コンパイル部１１０においてソースファイル群１０は機械語の中間コード群に変換され（ステップＳ１００１）、次に選択候補テーブル１３１が作成される（ステップＳ１００２）。

続いて、リンク部１４０において中間コード群がリンクされ実行可能な機械語コードが生成され（ステップＳ１００３）、関数選択部１４４において圧縮対象となる関数が選択される（ステップＳ１００４）。最後に、圧縮・伸張ビット付加部が機械語コード中の、圧縮対象の関数のコール命令に圧縮制御ビット、及びリターン命令に伸張制御ビットを付加し実行ファイルとして出力する（ステップＳ１００５）。

【００５９】

次に、プログラム変換装置１００の動作中で選択候補テーブルを作成する動作（ステップＳ１００２）を、図を用いて説明する。

図２２は、コンパイル部１１０において選択候補テーブルを作成する動作の流れを示す図である。

先ず、ある関数を調査対象とし調査を開始する（ステップＳ１１０１）。スタックアクセス検出部１１３が、対象の関数内で関数先頭でのスタックポインタが示すアドレスより上位のアドレスのスタックメモリへのアクセスがあるか否かを判定する（ステップＳ１１０２）。スタックアクセス検出部１１３が、対象の関数内でアクセスがあると判定する場合（ステップＳ１１０２：Yes）、対象の関数の関数評価値１３１ｂに０を設定する（ステップＳ１１０３）。

【００６０】

一方、スタックアクセス検出部１１３が、対象の関数内でアクセスがないと判定する場合（ステップＳ１１０２：No）、ユーザ指定関数抽出部１１４が、対象

の関数内にプラグマ#STACK#COMPRESSが記述されているか否かを判定する（ステップＳ１１０４）。ユーザ指定関数抽出部１１４が、対象の関数内にプラグマ#STACK#COMPRESSが記述されていると判定する場合（ステップＳ１１０４：Yes）、対象の関数の関数評価値１３１ｂに２を設定し（ステップＳ１１０５）、ユーザ指定関数抽出部１１４が、対象の関数内にプラグマ#STACK#COMPRESSが記述されていないと判定する場合（ステップＳ１１０４：No）、対象の関数の関数評価値１３１ｂに１を設定する（ステップＳ１１０６）。

【００６１】

上記動作で対象の関数の関数評価値１３１ｂが設定された後、さらに未調査の関数が存在する場合（ステップＳ１１０７：Yes）、未調査の関数を新たな調査対象とし調査を開始する（ステップＳ１１０１）。未調査の関数が存在しない場合（ステップＳ１１０７：No）、選択候補テーブルの作成が終了する。

次に、プログラム変換装置１００の動作中で圧縮対象となる関数を選択する動作（ステップＳ１００４）を、図を用いて説明する。

【００６２】

図２３は、リンク部１４０において圧縮対象となる関数を選択する動作の流れを示す図である。

先ず、ある関数を調査対象とし調査を開始する（ステップＳ１２０１）。関数選択部１４４が、対象の関数の関数評価値１３１ｂに２が設定されているか否かを判定する（ステップＳ１２０２）。関数選択部１４４が、対象の関数の関数評価値１３１ｂに２が設定されていると判定する場合（ステップＳ１２０２：Yes）、対象の関数が圧縮対象となる関数に選択される（ステップＳ１２０７）。

【００６３】

一方、関数選択部１４４が、対象の関数の関数評価値１３１ｂに２が設定されていないと判定する場合（ステップＳ１２０２：No）、対象の関数がツリーの先頭関数となるツリーがあるか否かを判定する（ステップＳ１２０３）、対象の関数がツリーの先頭関数となるツリーがある場合（ステップＳ１２０３：Yes）、対象の関数がツリーの先頭関数となるツリーを調査対象とし（ステップＳ１２０４）、調査対象のツリーのツリー評価値１４３ｃに、０が設定されているか否か

を判定する（ステップＳ１２０５）。調査対象のツリーのツリー評価値１４３ｃに０が設定されていない場合（ステップＳ１２０５：No）、対象の関数がツリーの先頭関数となる未調査のツリーが他にあるか否かを判定する（ステップＳ１２０６）。対象の関数がツリーの先頭関数となる未調査のツリーが他にある場合（ステップＳ１２０６：Yes）、未調査のツリーを調査対象とする（ステップＳ１２０４）。対象の関数がツリーの先頭関数となる未調査のツリーが他にない場合（ステップＳ１２０６：No）、対象の関数が圧縮対象となる関数に選択される（ステップＳ１２０７）。

【００６４】

対象の関数がツリーの先頭関数となるツリーがない場合（ステップＳ１２０３：No）、及び調査対象のツリーのツリー評価値１４３ｃに、０が設定されている場合（ステップＳ１２０５：Yes）、または上記動作で対象の関数が圧縮対象となる関数に選択された後、さらに未調査の関数が存在する場合（ステップＳ１２０８：Yes）、未調査の関数を新たな調査対象とし調査を開始する（ステップＳ１２０１）。未調査の関数が存在しない場合（ステップＳ１２０８：No）、圧縮対象となる関数の選択が終了する。

（情報処理装置）

ここで、上述のように構成された情報処理装置２００の動作について説明する。

【００６５】

情報処理装置２００は、処理を複数のステージに分けるパイプラインを採用することにより機械語命令を高速に実行するが、ここでは一つの機械語命令について処理の各ステージを通した動作を説明する。

図２４は、情報処理装置２００の動作の流れを示す図である。

先ず、ＩＦ２２１が、ＰＣ２２６が示す命令メモリ２１０上のアドレスから機械語命令をフェッチする（ステップＳ２００１）。次に、ＤＥＣ２２２が、機械語命令の種類を判定する（ステップＳ２００２）。

【００６６】

ＤＥＣ２２２が、機械語命令をコール命令であると判定する場合（ステップＳ２００２：CALL）、関数の呼び出しと、それに伴う保証レジスタ２２５ａの退避が実行される（ステップＳ２００３）。

ＤＥＣ２２２が、機械語命令をリターン命令であると判定する場合（ステップＳ２００２：RET）、関数からの復帰と、それに伴う保証レジスタ２２５ａの復帰が実行される（ステップＳ２００４）。

【００６７】

ＤＥＣ２２２が、機械語命令をコール命令、及びリターン命令以外であると判定する場合（ステップＳ２００２：else）、実行部２２４が機械語命令を実行する（ステップＳ２００５）。

以上の動作により、一つの機械語命令の処理が終了する。

次に、情報処理装置２００の動作中で関数の呼び出しと、それに伴い保証レジスタ２２５ａを退避する動作（ステップＳ２００３）を、図を用いて説明する。

【００６８】

図２５は、関数の呼び出しと、それに伴い保証レジスタ２２５ａを退避する動作の流れを示す図である。

先ず、ＤＥＣ２２２が、コール命令に圧縮制御ビットが付加されているか否かを判定する（ステップＳ２１０１）。ＤＥＣ２２２が、コール命令に圧縮制御ビットが付加されていないと判定する場合（ステップＳ２１０１：No）、制御部２２３が、ＰＣ２２６のデータをスタックメモリ２３１に格納し（ステップＳ２１０２）、続いて、制御部２２３が、保証レジスタ２２５ａの２５６ビットのデータをスタックメモリ２３１へ格納する（ステップＳ２１０３）。

【００６９】

一方、ＤＥＣ２２２が、コール命令に圧縮制御ビットが付加されていると判定する場合（ステップＳ２１０１：Yes）、制御部２２３が、ＰＣ２２６のデータをスタックメモリ２３１に格納し（ステップＳ２１０４）、続いて、データ圧縮部２４１が、保証レジスタ２２５ａのデータをハフマン符号化し（ステップＳ２１０５）、圧縮比較部２４３が、ハフマン符号化したデータのサイズがしきい値より小さいか否かを判定する（ステップＳ２１０６）、圧縮比較部２４３が、ハ

フマン符号化したデータのサイズがしきい値より小さいと判定する場合（ステップＳ２１０６：Yse）、圧縮制御部２４２が、ハフマン符号化したデータをスタックメモリ２３１へ格納し（ステップＳ２１０７）、続いて、圧縮制御部２４２が、ＣＳＲ２５０のイネーブルビットに１を、データフィールドにハフマン符号化したデータのサイズを設定する（ステップＳ２１０８）。

【００７０】

圧縮比較部２４３が、ハフマン符号化したデータのサイズがしきい値より小さくないと判定する場合（ステップＳ２１０６：No）、圧縮制御部２４２が、保証レジスタ２２５ａの２５６ビットのデータをそのままスタックメモリ２３１へ格納し（ステップＳ２１０９）、続いて、圧縮制御部２４２が、ＣＳＲ２５０のイネーブルビットに０を設定する（ステップＳ２１１０）。

【００７１】

ＣＳＲ２５０の設定後、圧縮制御部２４２が、ＣＳＲ２５０のデータをスタックメモリ２３１へ格納する（ステップＳ２１１１）。

最後に、呼び出し先の関数へ処理を分岐する（ステップＳ２１１２）。

次に、情報処理装置２００の動作中で関数からの復帰と、それに伴い保証レジスタ２２５ａを復帰する動作（ステップＳ２１０４）を、図を用いて説明する。

【００７２】

図２６は、関数からの復帰と、それに伴い保証レジスタ２２５ａを復帰する動作の流れを示す図である。

先ず、ＤＥＣ２２２が、リターン命令に伸張制御ビットが付加されているか否かを判定する（ステップＳ２２０１）。ＤＥＣ２２２が、リターン命令に伸張制御ビットが付加されていないと判定する場合（ステップＳ２２０１：No）、制御部２２３が、スタックメモリ２３１から保証レジスタ２２５ａへ２５６ビットの退避データを読み出す（ステップＳ２２０２）。

【００７３】

一方、ＤＥＣ２２２が、リターン命令に伸張制御ビットが付加されていると判定する場合（ステップＳ２２０１：Yes）、伸張制御部２６１が、スタックメモリ２３１からＣＳＲ２５０へ退避データを読み出し（ステップＳ２２０３）、続

いて、伸張制御部２６１が、読み出したＣＳＲ２５０のイネーブルビットが１であるか否かを判定する（ステップＳ２２０４）。伸張制御部２６１が、読み出したＣＳＲ２５０のイネーブルビットが１でないと判定する場合（ステップＳ２２０４：No）、伸張制御部２６１が、スタックメモリ２３１から保証レジスタ２２５ａへ２５６ビットの退避データをそのまま読み出す（ステップＳ２２０２）。

【００７４】

伸張制御部２６１が、読み出したＣＳＲ２５０のイネーブルビットが１であると判定する場合（ステップＳ２２０４：Yes）、伸張制御部２６１が、スタックメモリ２３１からＣＳＲ２５０のデータフィールドが示すサイズのハフマン符号化したデータを読み出し（ステップＳ２２０５）、データ伸張部２６２が、ハフマン符号化したデータを逆変換しデータを伸張し（ステップＳ２２０６）、伸張制御部２６１が、保証レジスタ２２５ａへ２５６ビットに伸張されたデータを復帰する（ステップＳ２２０７）。

【００７５】

以上の動作により保証レジスタ２２５ａのデータが復帰した後、制御部２２３が、スタックメモリ２３１からＰＣ２２６へ退避データを読み出し（ステップＳ２２０８）、最後に、ＰＣ２２６が示す呼び出し元の関数へ処理を分岐する（ステップＳ２２０９）。

＜まとめ＞

本発明の実施の形態は、関数の呼び出し時にスタックメモリへ退避する保証レジスタの内容を圧縮しスタックメモリの使用効率を上げることにより、スタックメモリのオーバーフローが発生する危険性を抑え、省メモリを実現するものであり、全ての関数の呼び出し時にスタックメモリへ退避するレジスタの内容を圧縮した場合、スタックメモリの使用効率は上がるが関数の呼び出しのたびに圧縮動作が発生しサイクル数が増加することになるので、スタックメモリに格納される期間が長い関数、例えばツリー構造になった関数群の先頭の関数、及びユーザにより指定された関数を呼び出し時にレジスタ内容を圧縮する対象に選択することでサイクル数の増加を抑えつつスタックメモリの使用効率を上げることを実現している。

【００７６】

【発明の効果】

本発明に係る情報処理装置は、演算に用いるデータを保持する１つ以上のレジスタを有し、機械語プログラムに従って処理を行うプロセッサを含む情報処理装置であって、所定の関数の呼び出しに伴いスタックメモリへ前記レジスタの内容を圧縮して退避することを示す情報が、前記機械語プログラム中に有るか否かを識別する圧縮識別手段と、圧縮識別手段による識別結果が肯定的である場合には所定の関数の呼び出しに伴いスタックメモリへ前記レジスタの内容を圧縮して退避し、圧縮識別手段による識別結果が否定的である場合には所定の関数の呼び出しに伴いスタックメモリへ前記レジスタの内容を圧縮せずに退避する退避手段とを備えることを特徴とする。

【００７７】

これによって、関数の呼び出しに伴いスタックメモリへ退避するレジスタの内容を圧縮することができるので、スタックメモリに格納するデータ量を削減することができる。

従って、スタックメモリの使用効率を上げることでスタックメモリのオーバーフローが発生する危険性を抑え、省メモリを実現することができる。

【００７８】

また、前記情報処理装置はさらに、所定の関数の呼び出し終了に伴い前記レジスタへスタックメモリに退避されているデータを伸張して復帰することを示す情報が、前記機械語プログラム中に有るか否かを識別する伸張識別手段と、伸張識別手段による識別結果が肯定的である場合には所定の関数の呼び出し終了に伴い前記レジスタへ前記スタックメモリに退避されているデータを伸張して復帰し、伸張識別手段による識別結果が否定的である場合には所定の関数の呼び出し終了に伴い前記レジスタへ前記スタックメモリに退避されているデータを伸張せずに復帰する復帰手段とを備えることを特徴とする。

【００７９】

あるいは、前記退避手段は、所定の関数の呼び出しに伴いスタックメモリへ前記レジスタの内容を圧縮して退避する場合に、所定の関数の呼び出し終了に伴い

前記レジスタへスタックメモリに退避されているデータを伸張して復帰することを示す伸張情報と、前記レジスタの内容を圧縮したデータとを対応付けてスタックメモリに退避する伸張情報退避手段を含み、前記情報処理装置はさらに、所定の関数の呼び出し終了に伴い前記レジスタへ復帰しようとするデータと対応付けられている伸張情報が、スタックメモリ中に有るか否かを識別する伸張識別手段と、伸張識別手段による識別結果が肯定的である場合には所定の関数の呼び出し終了に伴い前記レジスタへスタックメモリ中の前記復帰しようとするデータを伸張して復帰し、伸張識別手段による識別結果が否定的である場合には所定の関数の呼び出し終了に伴い前記レジスタへスタックメモリ中の前記復帰しようとするデータを伸張せずに復帰する復帰手段とを備えることを特徴とする。

【００８０】

これによって、関数の呼び出し終了に伴いレジスタへ復帰するスタックメモリ中の圧縮された内容を伸張することができるので、関数の呼び出しの前後でのレジスタの値を保証したうえで、スタックメモリに格納するデータ量を削減することができる。

従って、スタックメモリの使用効率を上げることでスタックメモリのオーバーフローが発生する危険性を抑え、省メモリを実現することができる。

【００８１】

また、前記退避手段はさらに、前記レジスタに格納されている第１データを所定のアルゴリズムに従って、第２データに変換するデータ変換手段と、前記第２データのデータサイズを、圧縮効率を示すしきい値と比較する比較手段と、前記第２データのデータサイズがしきい値より小さい場合には前記第２データをスタックメモリへ退避し、前記第２データのデータサイズがしきい値より大きい場合には前記第１データをスタックメモリへ退避する選択退避手段とを含むことを特徴とする。

【００８２】

これによって、保証レジスタのデータに圧縮を施し、圧縮効果が有る場合のみスタックメモリに圧縮したデータを退避し、圧縮効果がない場合は保証レジスタのデータをそのままスタックメモリに退避することができる。

従って、保証レジスタの各ビットのデータに相関が少なく、圧縮効率の悪い場合に、ハフマン符号化等のデータ変換によるデータサイズの増大の影響を受けることなくスタックメモリの使用効率を上げることを実現できる。

【００８３】

また、前記退避手段は、圧縮識別手段による識別結果が肯定的である場合、前記所定の関数の呼び出し元における呼び出し時にスタックメモリへ前記レジスタの内容を圧縮して退避し、前記復帰手段は、伸張識別手段による識別結果が肯定的である場合、前記所定の関数の呼び出し元における呼び出し終了時に前記レジスタへスタックメモリに退避されているデータを伸張して復帰することを特徴とする。

【００８４】

これによって、コール命令の動作時に、スタックメモリへ退避する保証レジスタの内容を圧縮し、リターン命令の動作時に、保証レジスタへ復帰するスタックメモリの内容を伸張することができる。

従って、関数への分岐命令がスタックメモリへの保証レジスタの退避を伴う情報処理装置において、関数の前後でのレジスタの値を保証したうえでスタックメモリに格納するデータ量を削減し、スタックメモリの使用効率を上げることができる。

【００８５】

また、前記退避手段は、圧縮識別手段による識別結果が肯定的である場合、呼び出された前記所定の関数における処理開始時にスタックメモリへ前記レジスタの内容を圧縮して退避し、前記復帰手段は、伸張識別手段による識別結果が肯定的である場合、呼び出された前記所定の関数における処理終了時に前記レジスタへスタックメモリに退避されているデータを伸張して復帰することを特徴とする。

【００８６】

これによって、ストア命令による保証レジスタのスタックメモリへの退避時に、保証レジスタの内容を圧縮し、ロード命令によるスタックメモリの保証レジスタへの復帰時に、スタックメモリの内容を伸張することができる。

従って、関数への分岐命令がスタックメモリへの保証レジスタの退避を伴わない情報処理装置において、関数の前後でのレジスタの値を保証したうえでスタックメモリに格納するデータ量を削減し、スタックメモリの使用効率を上げることができる。

【００８７】

また、演算に用いるデータを保持する１つ以上のレジスタを有し、機械語プログラムに従って処理を行うプロセッサを含む情報処理装置における情報処理方法であって、所定の関数の呼び出しに伴いスタックメモリへ前記レジスタの内容を圧縮して退避することを示す情報が、前記機械語プログラム中に有るか否かを識別する圧縮識別ステップと、圧縮識別ステップによる識別結果が肯定的である場合には所定の関数の呼び出しに伴いスタックメモリへ前記レジスタの内容を圧縮して退避し、圧縮識別ステップによる識別結果が否定的である場合には所定の関数の呼び出しに伴いスタックメモリへ前記レジスタの内容を圧縮せずに退避する退避ステップとを備えることを特徴とする。

【００８８】

これによって、関数の呼び出しに伴いスタックメモリへ退避するレジスタの内容を圧縮することができるので、スタックメモリに格納するデータ量を削減することができる。

従って、スタックメモリの使用効率を上げることでスタックメモリのオーバーフローが発生する危険性を抑え、省メモリを実現することができる。

【００８９】

また、プログラム変換装置であって、１、又は複数の関数を含む入力プログラムを入手する入手手段と、所定の関数の呼び出しに伴いスタックメモリへプロセッサ内のレジスタの内容を圧縮して退避するか、圧縮せずにそのまま退避するかを前記入力プログラムにおいて判定する判定手段と、当該判定手段により圧縮して退避すると判定された場合、スタックメモリへ前記レジスタの内容を圧縮して退避することをプロセッサに示す情報を備える出力プログラムに、前記入力プログラムを変換する変換手段とを備えることを特徴とする。

【００９０】

これによって、全ての関数の呼び出しを保証レジスタの圧縮対象とせず、保証レジスタを圧縮する関数を選択的に決定することができる。

従って、全ての関数で保証レジスタを圧縮することによるサイクル数の大幅な増加をさけつつ、スタックメモリの使用効率を上げることを実現することができる。

【００９１】

また、前記判定手段は、前記レジスタの内容を格納したスタックメモリを参照するスタックアクセス関数を検出する検出手段と、前記スタックアクセス関数、及び前記スタックアクセス関数の呼び出し側の上位に位置する全ての階層の関数の呼び出しに伴いスタックメモリへ前記レジスタの内容を圧縮せずにそのまま退避すると判定する除外手段とを含むことを特徴とする。

【００９２】

これによって、関数の処理開始時のスタックポインタが示すアドレスより上位のアドレスのスタックメモリにアクセスする関数を、ツリー構造の下位に持つ関数を保証レジスタの内容を圧縮する対象から除くことができる。

従って、すでにスタックメモリに格納されているデータを参照する場合に、参照するデータが圧縮されていることを避けることができる。

【００９３】

また、前記判定手段は、スタックメモリへ前記レジスタの内容を圧縮して退避することを示す情報が前記所定の関数に事前に付加されている場合、前記所定の関数の呼び出しに伴いスタックメモリへ前記レジスタの内容を圧縮して退避すると判定する事前指定判定手段を含むことを特徴とする。

これによって、変換前のプログラム中で事前に指定された関数を、変換後のプログラムにおいて、保証レジスタの圧縮対象とすることができる。

【００９４】

従って、プログラム時（コンパイル前）に、プログラマ（ユーザ）が所望の関数を指定し、保証レジスタの圧縮対象を静的に決定することができる。

また、前記判定手段は、前記所定の関数が内部にサブルーチンを持つ場合、前記入力プログラムが含む関数群の階層構造を示すネスト情報に基づいて、前記所

定の関数の呼び出しに伴いスタックメモリへ前記レジスタの内容を圧縮して退避するか、圧縮せずにそのまま退避するかを判定するネスト判定手段を含むことを特徴とする。

【００９５】

これによって、ツリー構造をもつ関数群の任意の関数をコンパイル時に動的に保証レジスタの圧縮対象とすることできる。

従って、一度スタックメモリに退避すると、長期間スタックに留まるツリー構造の上位の関数を、保証レジスタの圧縮対象とすることで、スタックメモリの使用効率を上げることが実現できる。

【００９６】

また、前記変換手段は、前記所定の関数の呼び出しに伴いスタックメモリへ前記レジスタの内容を圧縮して退避することをプロセッサに示す情報を、前記所定の関数の呼び出し元関数に付加する圧縮情報付加手段と、前記所定の関数の呼び出し終了に伴い前記レジスタへスタックメモリに退避されているデータを伸張し復帰することをプロセッサに示す情報を、前記所定の関数の呼び出し元関数に付加する伸張情報付加手段とを含むことを特徴とする。

【００９７】

これによって、コール命令の動作時にスタックメモリへ退避する保証レジスタの内容を圧縮し、リターン命令の動作時に保証レジスタへ復帰するスタックメモリの内容を伸張することを示す情報をプログラム中に付加することができる。

従って、関数への分岐命令がスタックメモリへの保証レジスタの退避を伴う情報処理装置において、関数の前後でのレジスタの値を保証したうえでスタックメモリに格納するデータ量を削減し、スタックメモリの使用効率を上げるプログラムを生成することができる。

【００９８】

また、前記変換手段は、前記所定の関数の呼び出しに伴いスタックメモリへ前記レジスタの内容を圧縮して退避することをプロセッサに示す情報を、呼び出される前記所定の関数に付加する圧縮情報付加手段と、前記所定の関数の呼び出し終了に伴い前記レジスタへスタックメモリに退避されているデータを伸張し復帰

することをプロセッサに示す情報を、呼び出される前記所定の関数に付加する伸張情報付加手段とを含むことを特徴とする。

【００９９】

これによって、ストア命令による保証レジスタのスタックメモリへの退避時に保証レジスタの内容を圧縮し、ロード命令によるスタックメモリの保証レジスタへの復帰時にスタックメモリの内容を伸張することを示す情報をプログラム中に付加することができる。

従って、関数への分岐命令がスタックメモリへの保証レジスタの退避を伴わない情報処理装置において、関数の前後でのレジスタの値を保証したうえでスタックメモリに格納するデータ量を削減し、スタックメモリの使用効率を上げるプログラムを生成することができる。

【図面の簡単な説明】

【図１】

本発明の実施の形態に係わるプログラム変換装置、及び情報処理装置を備えた情報処理システムの構成を示す図である。

【図２】

ソースファイル１１の一例を模式的に示す図である。

【図３】

ソースファイル１２の一例を模式的に示す図である。

【図４】

本発明の実施の形態に係わるプログラム変換装置の構成を示す図である。

【図５】

選択候補情報保持部１３０が保持する選択候補テーブル１３１のデータ構造を示す図である。

【図６】

呼び出し関係にある関数のツリー構造を示す図である。

【図７】

関数ツリー情報保持部１４３で保持する関数ツリー情報テーブル１４３ａのデータ構造を示す図である。

【図８】

実行ファイル２０の内容を模式的に示す図である。

【図９】

分岐後にコーリー側でレジスタの退避、復帰を行う実行ファイルの内容を模式的に示す図である。

【図１０】

本発明の実施の形態に係わる情報処理装置の構成を示す図である。

【図１１】

処理部２２０の内部の構成を示す図である。

【図１２】

圧縮・伸張制御ビットが付加されているストア命令、ロード命令をデコードするＤＥＣ２２２の内部の構成を示す図である。

【図１３】

スタックメモリ２３１を模式的に示す図である。

【図１４】

保証レジスタ２２５ａのデータを退避したスタックメモリ２３１を模式的に示す図である。

【図１５】

圧縮部２４０の内部の構成を示す図である。

【図１６】

ハフマン符号対応表を示す図である。

【図１７】

図１４と同じ保証レジスタ２２５ａのデータを、図１６のハフマン符号対応表を基に圧縮して退避したスタックメモリ２３１を模式的に示す図である。

【図１８】

ＣＳＲ２５０のビット割付けを示す図である。

【図１９】

伸張部２６０の内部の構成を示す図である。

【図２０】

保証レジスタ２２５ａとスタックメモリ２３１間のデータ退避、復帰に伴うデータの流れを示す図である。

【図２１】

プログラム変換装置１００の動作の流れを示す図である。

【図２２】

コンパイル部１１０において選択候補テーブルを作成する動作の流れを示す図である。

【図２３】

リンク部１４０において圧縮対象となる関数を選択する動作の流れを示す図である。

【図２４】

情報処理装置２００の動作の流れを示す図である。

【図２５】

関数の呼び出しと、それに伴い保証レジスタ２２５ａを退避する動作の流れを示す図である。

【図２６】

関数からの復帰と、それに伴い保証レジスタ２２５ａを復帰する動作の流れを示す図である。

【符号の説明】

１０　ソースファイル群

１１　ソースファイル

１１ａ　関数mainに関する記述部

１１ｂ　関数func#aに関する記述部

１１ｃ　関数func#bに関する記述部

１１ｄ　関数func#dに関する記述部

１１ｅ　関数func#eに関する記述部

１１ｆ　プラグマ#STACK#COMPRESSの記述部

１１ｇ　関数func#fに関する記述部

１２　ソースファイル

１２ａ　関数func#cに関する記述部
２０　実行ファイル
１００　プログラム変換装置
１１０　コンパイル部
１１１　ソースファイル取得部
１１２　翻訳部
１１３　スタックアクセス検出部
１１４　ユーザ指定関数抽出部
１２０　中間コード保持部
１３０　選択候補情報保持部
１３１　選択候補テーブル
１３１ａ　関数名
１３１ｂ　関数評価値
１４０　リンク部
１４１　連結部
１４２　関数ツリー情報作成部
１４３　関数ツリー情報保持部
１４３ａ　関数ツリー情報テーブル
１４３ｂ　ツリー番号
１４３ｃ　ツリー評価値
１４４　関数選択部
１４５　圧縮・伸張ビット付加部
１５０　記憶部
２００　情報処理装置
２１０　命令メモリ
２２０　処理部
２２１　ＩＦ
２２２　ＤＥＣ
２２２ａ　圧縮制御ビット付きコール命令デコード部

２２２ｂ　伸張制御ビット付きリターン命令デコード部

２２２ｃ　圧縮制御ビット付きストア命令デコード部

２２２ｄ　伸張制御ビット付きロード命令デコード部

２２３　制御部

２２４　実行部

２２５　レジスタ

２２５ａ　保証レジスタ

２２６　ＰＣ

２２７　ＳＰ

２３０　データメモリ

２３１　スタックメモリ

２４０　圧縮部

２４１　データ圧縮部

２４２　圧縮制御部

２４３　圧縮比較部

２４４　しきい値保持部

２５０　ＣＳＲ

２６０　伸張部

２６１　伸張制御部

２６２　データ伸張部

３０１　関数func#aへ分岐するコール命令

３０２　関数func#eへ分岐するコール命令

３０３　関数func#aから復帰するリターン命令

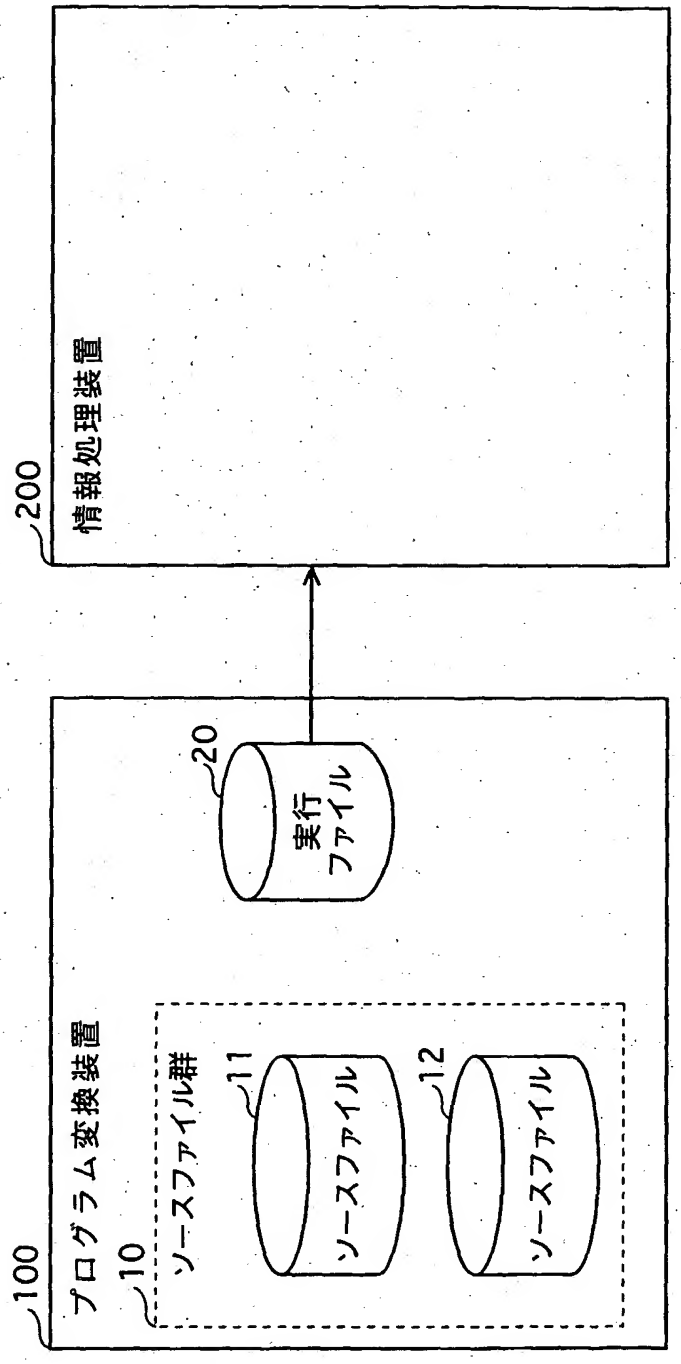
３０４　関数func#eから復帰するリターン命令

３０５　関数func#aの処理開始時におけるストア命令

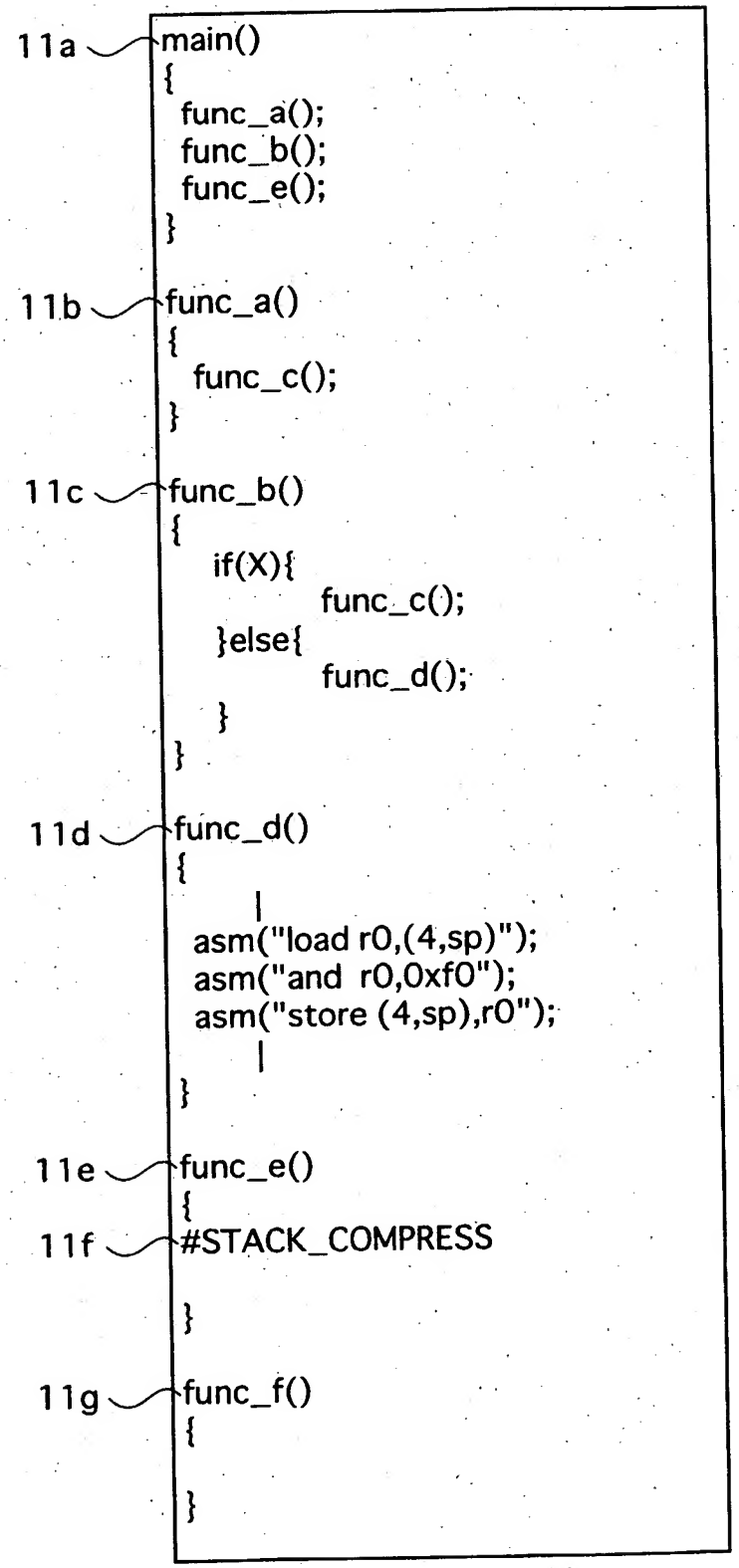
３０６　関数func#aの処理終了時におけるロード命令

【書類名】　　図面

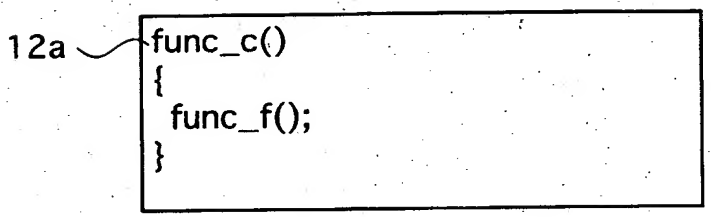
【図１】


200
情報処理装置
20
実行
ファイル
100
プログラム変換装置
10
ソースファイル群
11
ソースファイル
12
ソースファイル

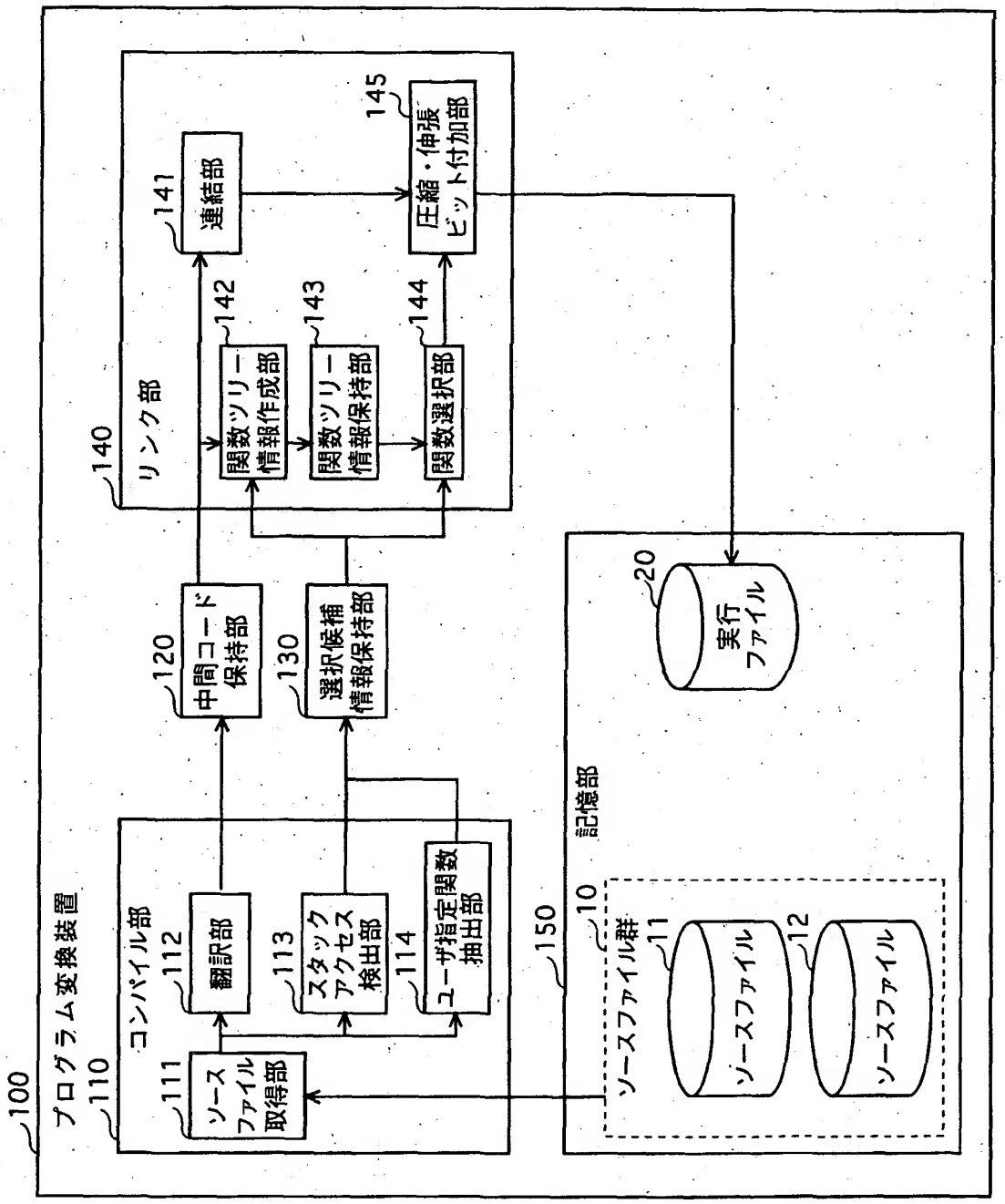
【図２】


11a
main()
{
func_a();
func_b();
func_e();
}
11b
func_a()
{
func_c();
}
11c
func_b()
{
if(X){
func_c();
}else{
func_d();
}
}
11d
func_d()
{
asm("load r0,(4,sp)");
asm("and r0,0xf0");
asm("store (4,sp),r0");
}
11e
func_e()
{
11f
#STACK_COMPRESS
}
11g
func_f()
{
}

【図３】



【図４】

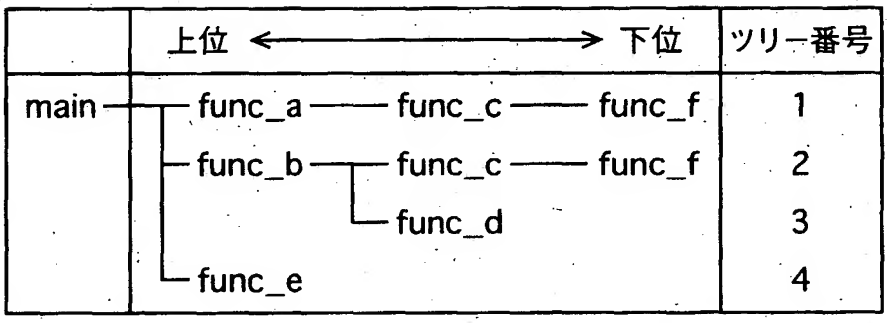

100
プログラム変換装置
110
コンパイル部
111
ソースファイル取得部
112
翻訳部
113
スタックアクセス検出部
114
ユーザ指定関数抽出部
120
中間コード保持部
130
選択候補情報保持部
140
リンク部
141
連結部
142
関数ツリー情報作成部
143
関数ツリー情報保持部
144
関数選択部
145
圧縮・伸張ビット付加部
150
記憶部
10
ソースファイル群
11
ソースファイル
12
ソースファイル
20
実行ファイル

【図５】

131　131a　131b

関数名	関数評価値
func_a	1
func_b	1
func_c	1
func_d	0
func_e	2
func_f	1

【図６】

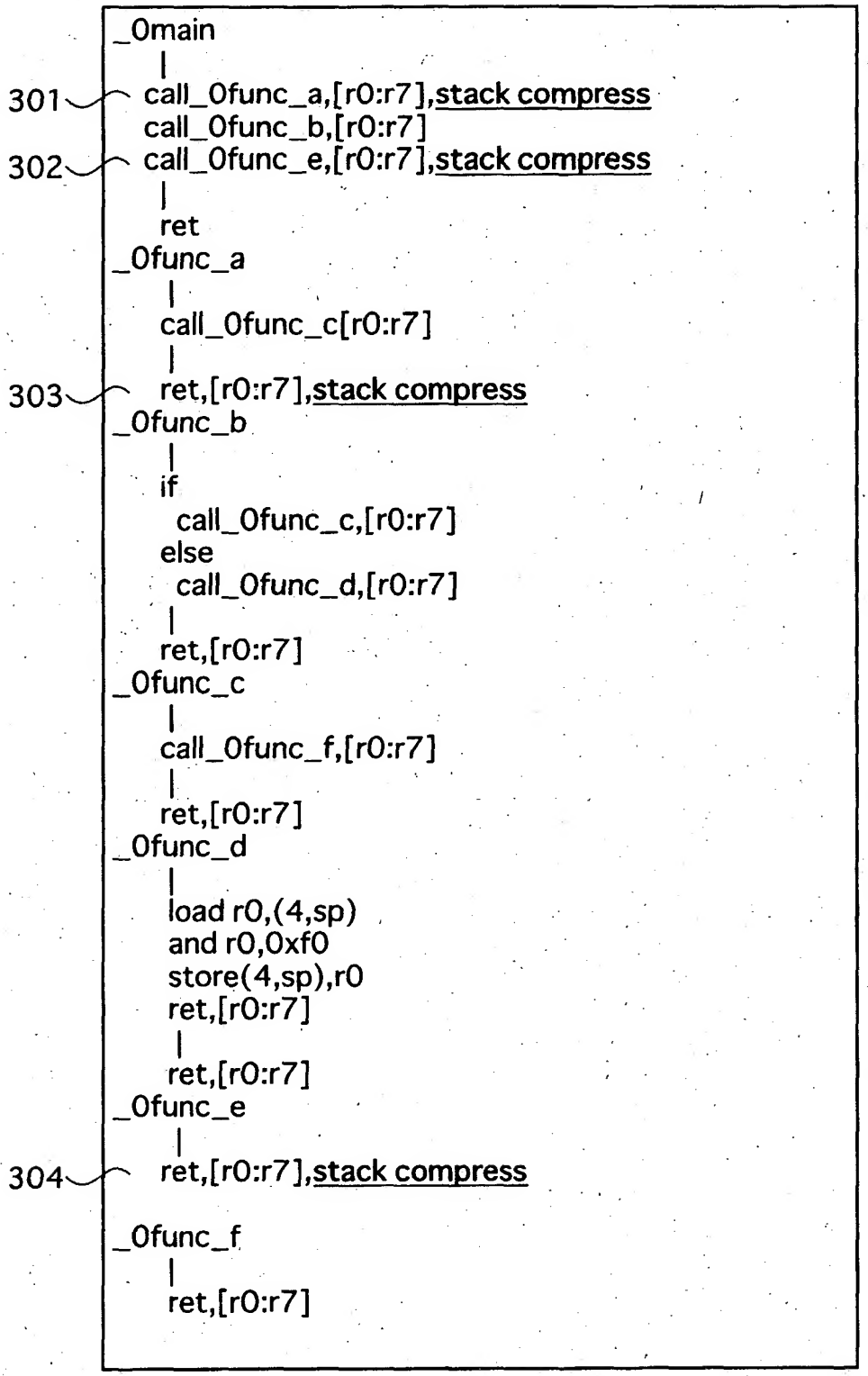

上位
下位
ツリー番号
main
func_a
func_c
func_f
1
func_b
func_c
func_f
2
func_d
3
func_e
4

【図７】

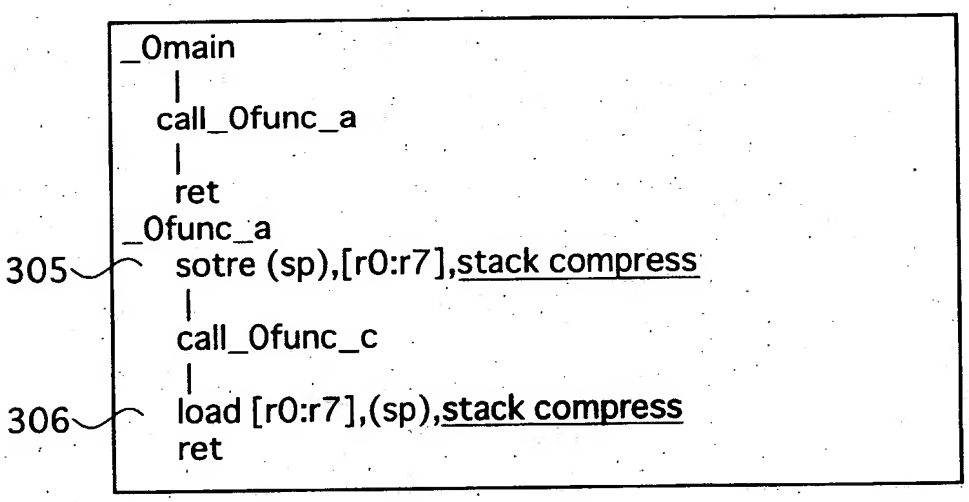
143a　143b　143c

ツリー番号	ツリー評価値
1	1 (=1×1×1)
2	1 (=1×1×1)
3	0 (=1×0)
4	2 (=2)

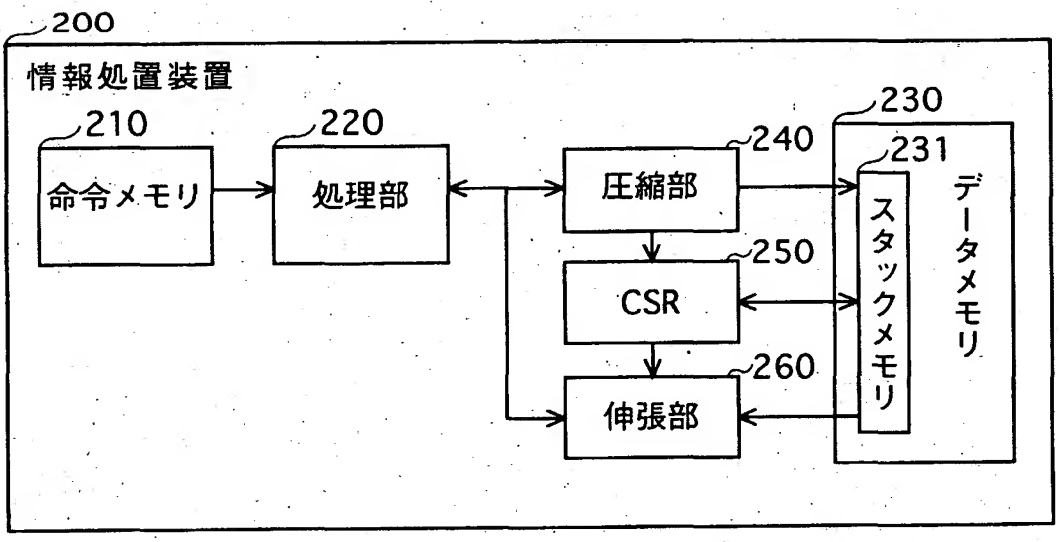
【図８】


_0main
call_0func_a,[r0:r7],stack compress
301
call_0func_b,[r0:r7]
call_0func_e,[r0:r7],stack compress
302
ret
_0func_a
call_0func_c[r0:r7]
ret,[r0:r7],stack compress
303
_0func_b
if
call_0func_c,[r0:r7]
else
call_0func_d,[r0:r7]
ret,[r0:r7]
_0func_c
call_0func_f,[r0:r7]
ret,[r0:r7]
_0func_d
load r0,(4,sp)
and r0,0xf0
store(4,sp),r0
ret,[r0:r7]
ret,[r0:r7]
_0func_e
ret,[r0:r7],stack compress
304
_0func_f
ret,[r0:r7]

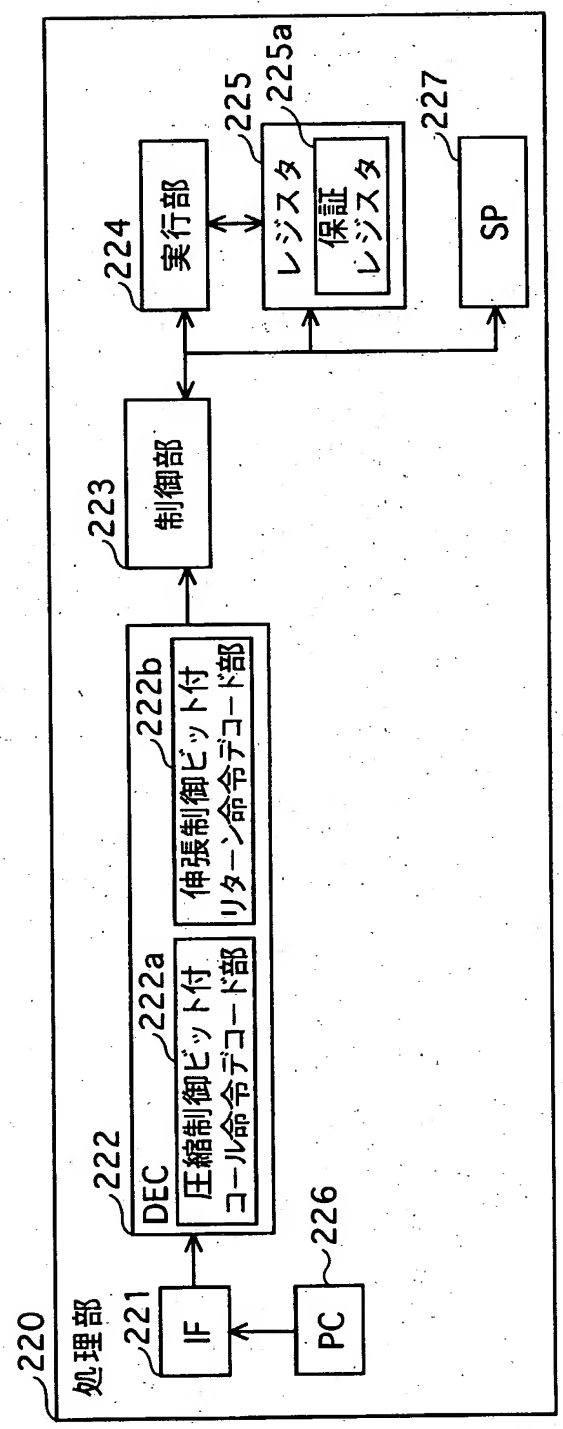
【図９】


_Omain
call_Ofunc_a
ret
_Ofunc_a
305
sotre (sp),[r0:r7],stack compress
call_Ofunc_c
306
load [r0:r7],(sp),stack compress
ret

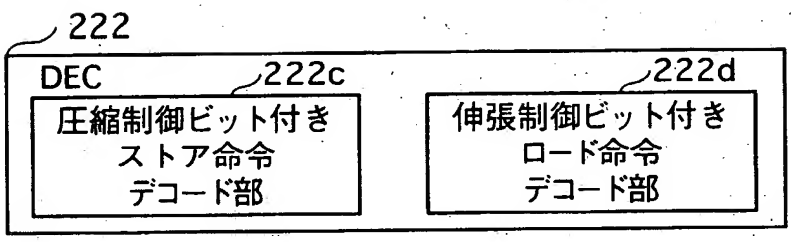
【図１０】


200
情報処置装置
210
命令メモリ
220
処理部
240
圧縮部
250
CSR
260
伸張部
230
231
スタックメモリ
データメモリ

【図１１】

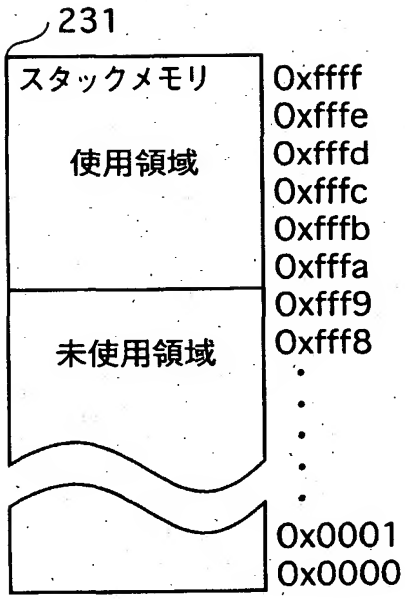

220
処理部
221
IF
PC
226
222
DEC
222a
圧縮制御ビット付
コール命令デコード部
222b
伸張制御ビット付
リターン命令デコード部
223
制御部
224
実行部
225
225a
レジスタ
保証
レジスタ
227
SP

【図１２】

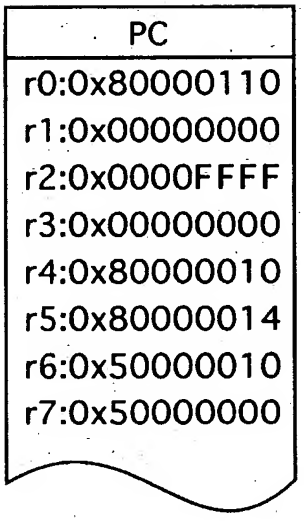


出証特２００３－３０１７２８２

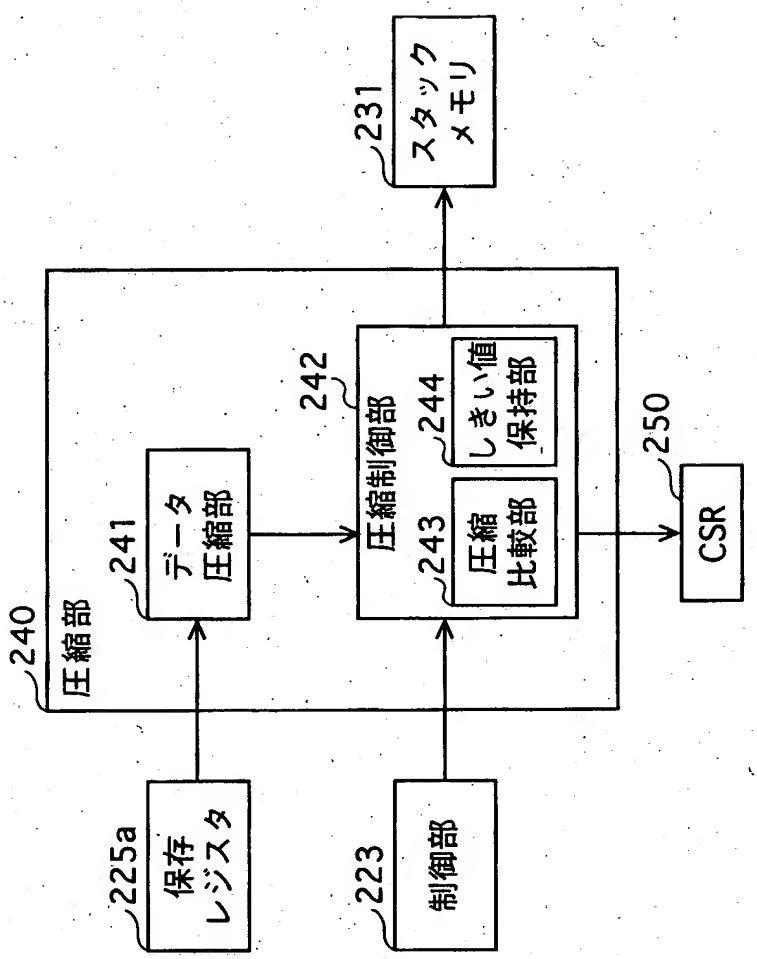
【図１３】


231
スタックメモリ
使用領域
未使用領域
0xffff
0xfffe
0xfffd
0xfffc
0xfffb
0xfffa
0xfff9
0xfff8
0x0001
0x0000

【図１４】


PC
r0:0x80000110
r1:0x00000000
r2:0x0000FFFF
r3:0x00000000
r4:0x80000010
r5:0x80000014
r6:0x50000010
r7:0x50000000

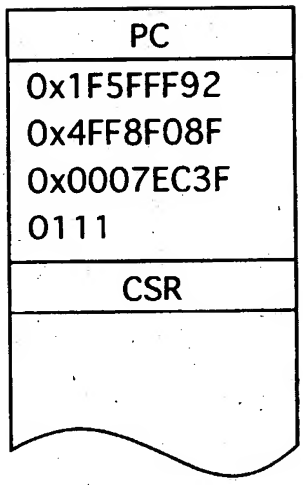
【図１５】


231
スタック
メモリ
242
圧縮制御部
244
しきい値
保持部
243
圧縮
比較部
250
CSR
241
データ
圧縮部
240
圧縮部
225a
保存
レジスタ
223
制御部

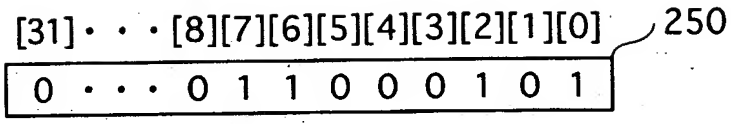
【図１６】

値	ハフマン符号
0x4	0b00000
0x5	0b00001
0x8	0b0001
0xf	0b001
0x1	0b01
0x0	0b1

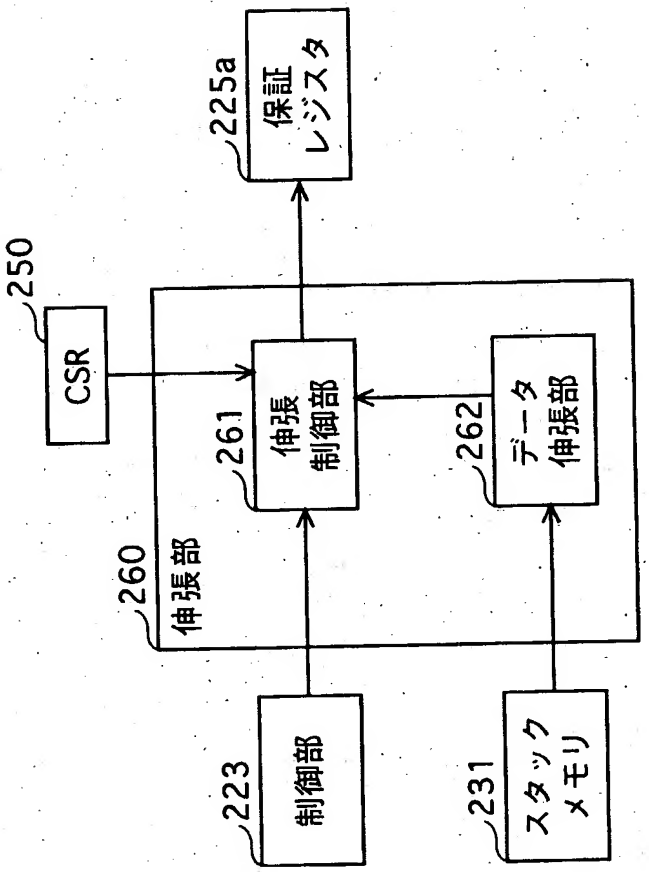
【図１７】



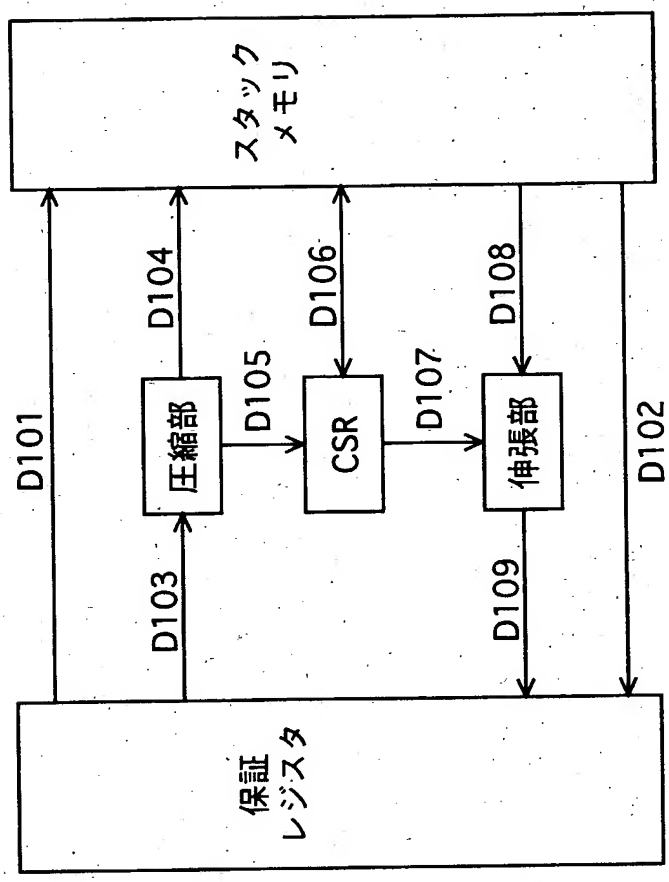
【図１８】



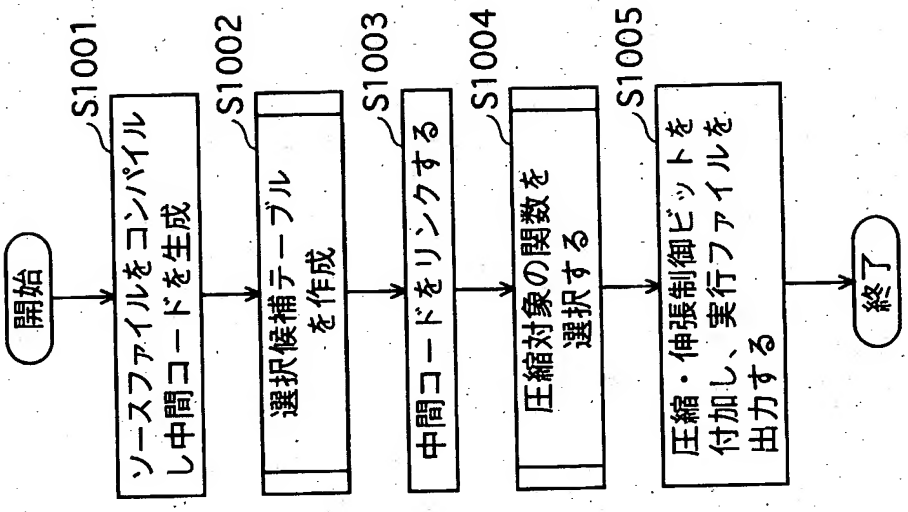
【図１９】


225a
保証
レジスタ
250
CSR
261
伸張
制御部
262
データ
伸張部
260
伸張部
223
制御部
231
スタック
メモリ

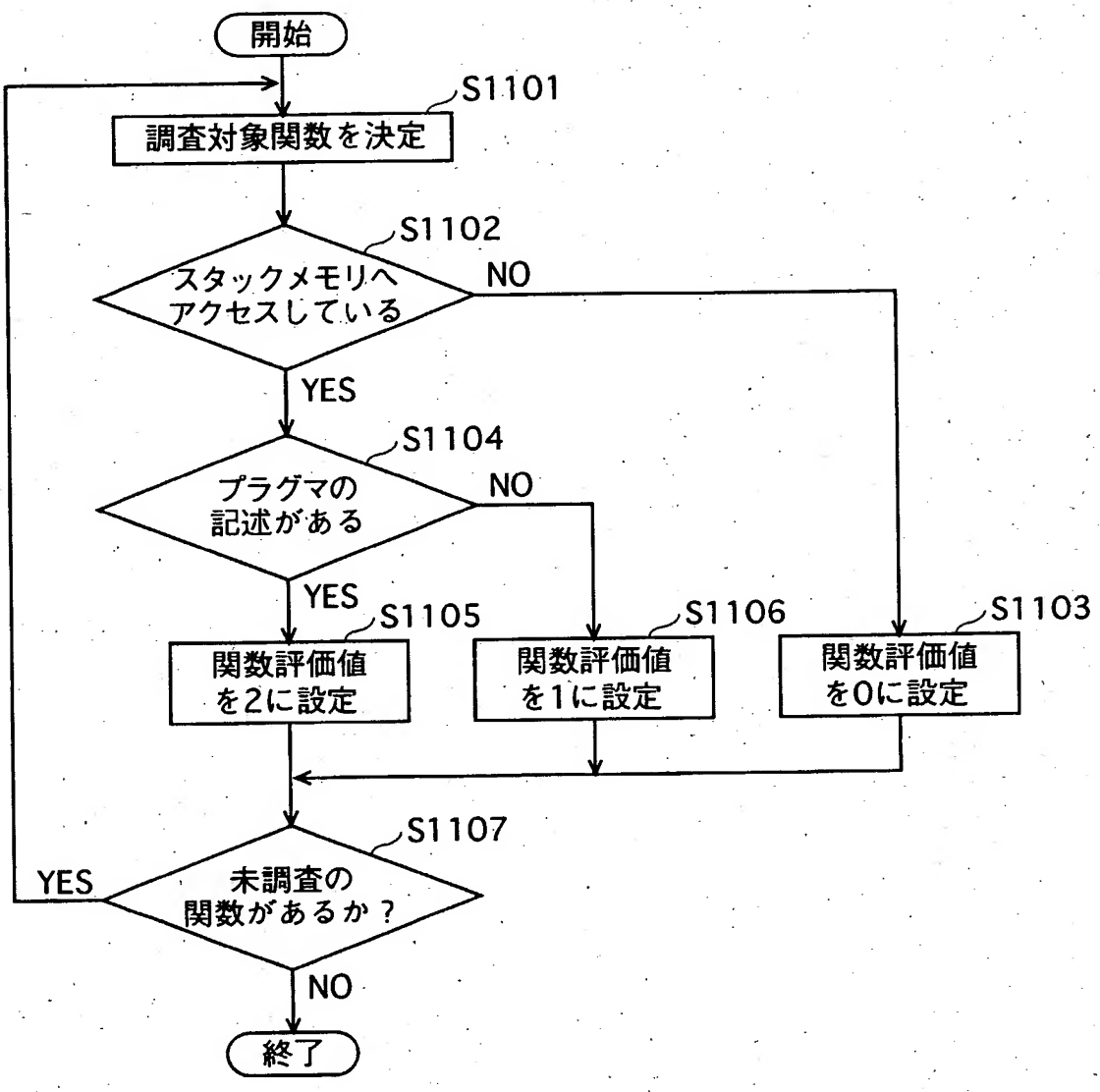
【図２０】


スタック
メモリ
D104
D106
D108
D101
D105
D107
D102
圧縮部
CSR
伸張部
D103
D109
保証
レジスタ

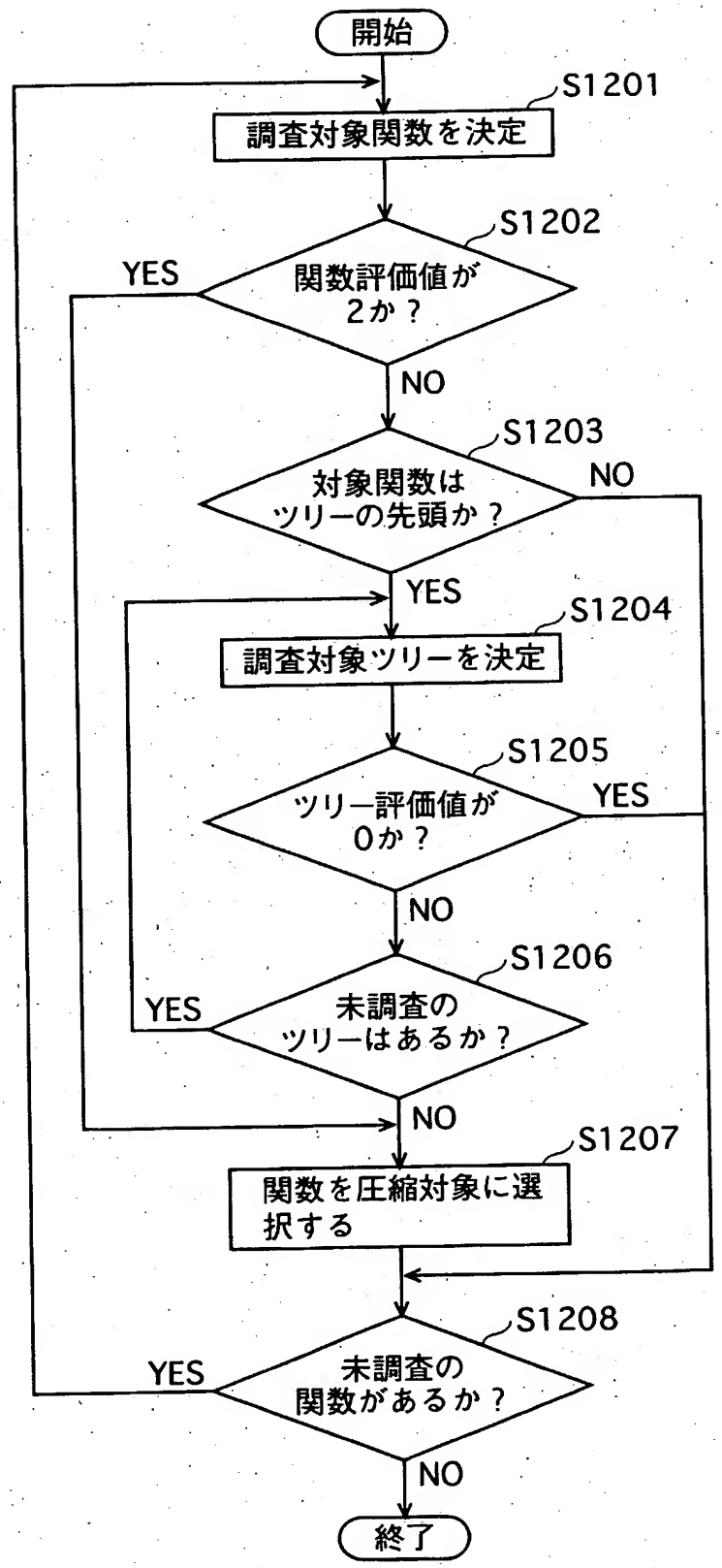
【図２１】


開始
S1001
ソースファイルをコンパイルし中間コードを生成
S1002
選択候補テーブルを作成
S1003
中間コードをリンクする
S1004
圧縮対象の関数を選択する
S1005
圧縮・伸張制御ビットを付加し、実行ファイルを出力する
終了

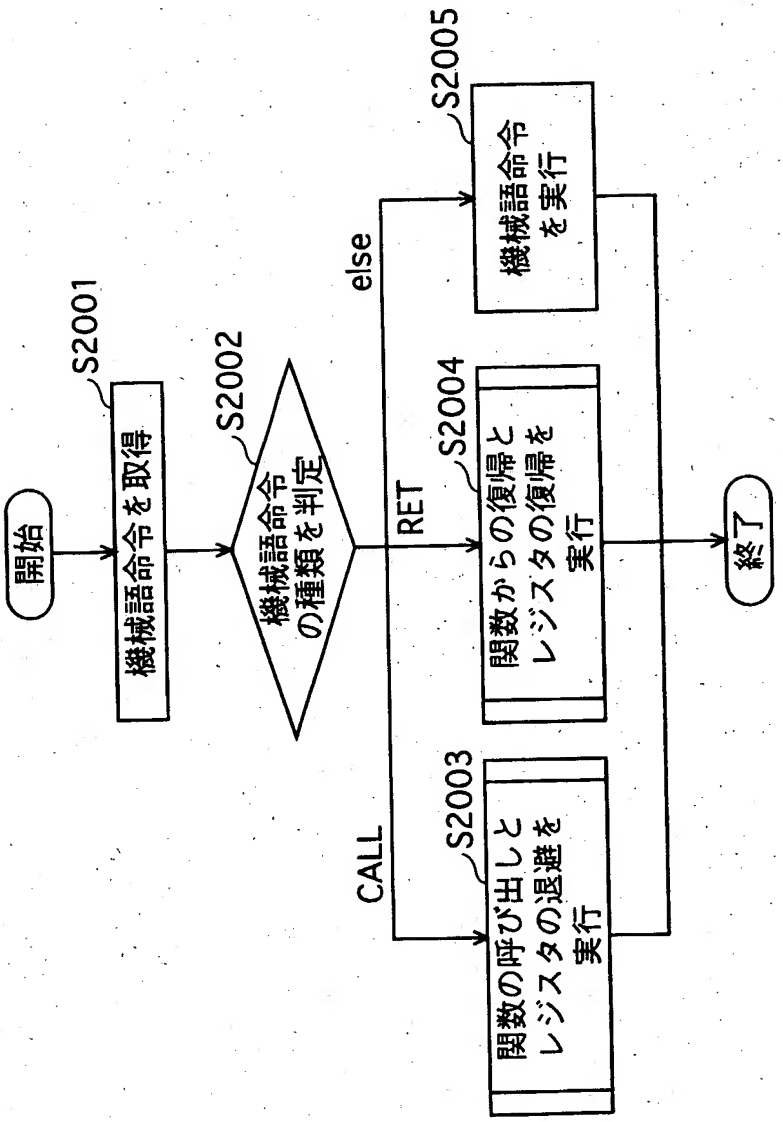
【図２２】


開始
S1101
調査対象関数を決定
S1102
スタックメモリへ
アクセスしている
NO
YES
S1104
プラグマの
記述がある
NO
YES
S1105
関数評価値
を2に設定
S1106
関数評価値
を1に設定
S1103
関数評価値
を0に設定
S1107
YES
未調査の
関数があるか？
NO
終了

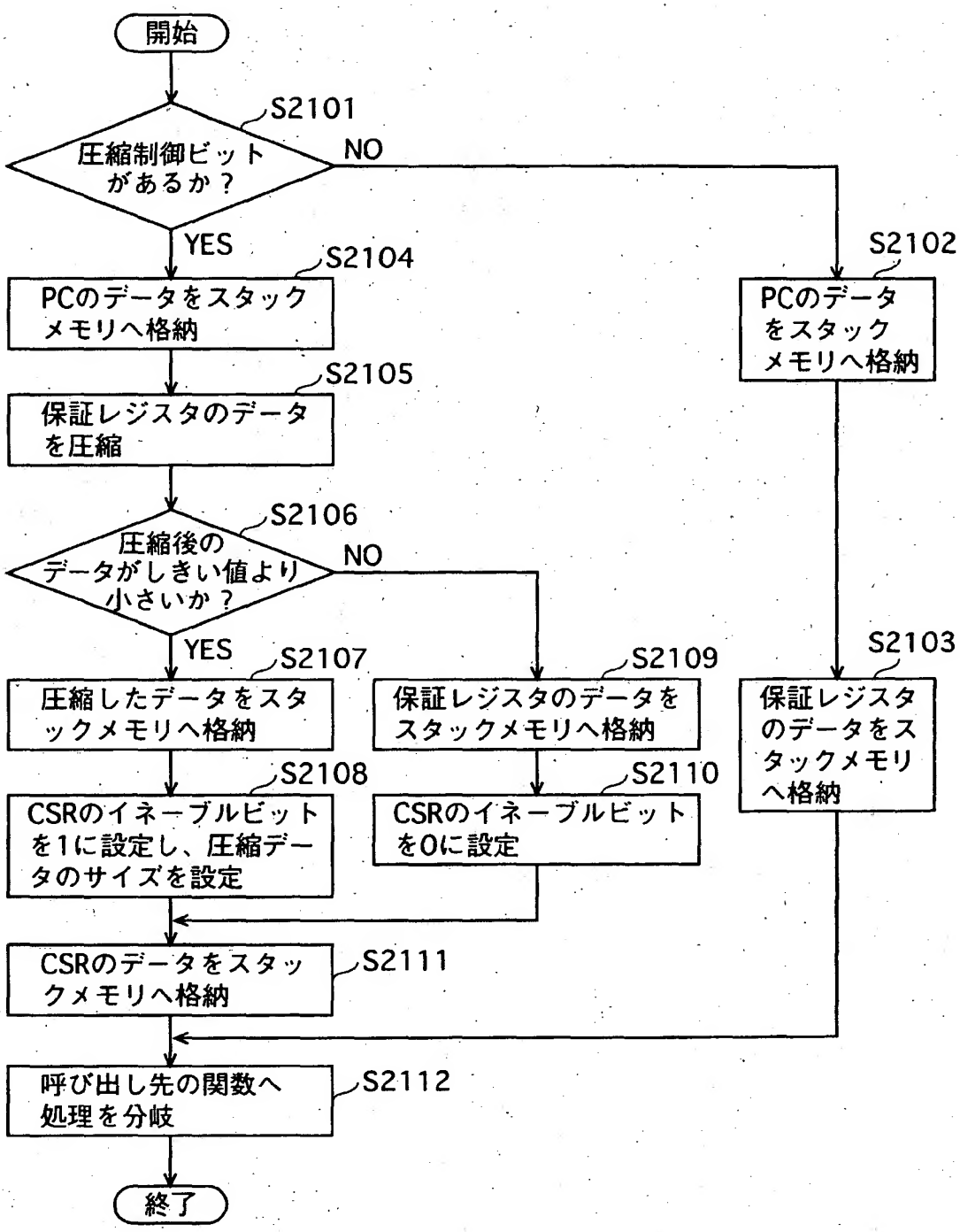
【図２３】


開始
S1201
調査対象関数を決定
S1202
関数評価値が
2か？
YES
NO
S1203
対象関数は
ツリーの先頭か？
NO
YES
S1204
調査対象ツリーを決定
S1205
ツリー評価値が
0か？
YES
NO
S1206
未調査の
ツリーはあるか？
YES
NO
S1207
関数を圧縮対象に選
択する
S1208
未調査の
関数があるか？
YES
NO
終了

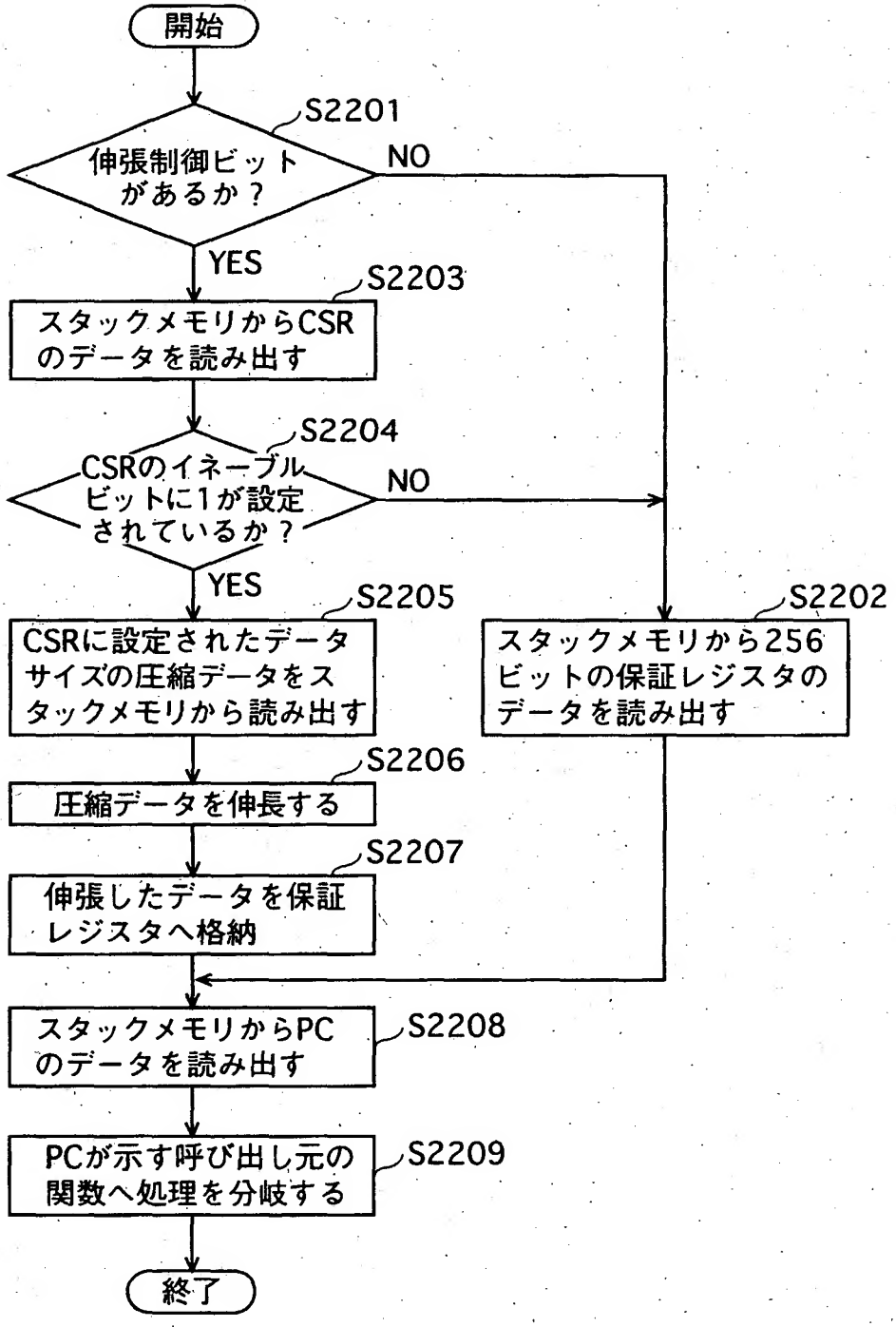
【図２４】


開始
S2001
機械語命令を取得
S2002
機械語命令の種類を判定
CALL
S2003
関数の呼び出しとレジスタの退避を実行
RET
S2004
関数からの復帰とレジスタの復帰を実行
else
S2005
機械語命令を実行
終了

【図２５】


開始
S2101
圧縮制御ビットがあるか？
NO
YES
S2104
PCのデータをスタックメモリへ格納
S2105
保証レジスタのデータを圧縮
S2106
圧縮後のデータがしきい値より小さいか？
NO
YES
S2107
圧縮したデータをスタックメモリへ格納
S2108
CSRのイネーブルビットを1に設定し、圧縮データのサイズを設定
S2109
保証レジスタのデータをスタックメモリへ格納
S2110
CSRのイネーブルビットを0に設定
S2111
CSRのデータをスタックメモリへ格納
S2112
呼び出し先の関数へ処理を分岐
終了
S2102
PCのデータをスタックメモリへ格納
S2103
保証レジスタのデータをスタックメモリへ格納

【図２６】


開始
S2201
伸張制御ビットがあるか？
NO
YES
S2203
スタックメモリからCSRのデータを読み出す
S2204
CSRのイネーブルビットに1が設定されているか？
NO
YES
S2205
CSRに設定されたデータサイズの圧縮データをスタックメモリから読み出す
S2202
スタックメモリから256ビットの保証レジスタのデータを読み出す
S2206
圧縮データを伸長する
S2207
伸張したデータを保証レジスタへ格納
スタックメモリからPCのデータを読み出す
S2208
PCが示す呼び出し元の関数へ処理を分岐する
S2209
終了

【書類名】　　要約書

【要約】

【課題】　情報処理装置においてスタックメモリの使用効率を高める。

【解決手段】　情報処理装置２００であって、所定の関数の呼び出しに伴いスタックメモリ２３１へ保証レジスタの内容を圧縮して退避することを示す圧縮情報が機械語プログラム中に有るか否かをデコーダにおいて識別し、圧縮情報が有る場合に保証レジスタの内容を圧縮してスタックメモリ２３１へ退避する圧縮部２４０と、所定の関数の呼び出し終了に伴い保証レジスタへスタックメモリ２３１の内容を伸張して復帰することを示す伸張情報が機械語プログラム中に有るか否かをデコーダにおいて識別し、伸張情報が有る場合に、スタックメモリ２３１の内容を伸張して保証レジスタへ復帰する伸張部２６０とを備える。

【選択図】　　図１０

出願人履歴情報

識別番号　〔０００００５８２１〕

１．変更年月日　１９９０年　８月２８日
［変更理由］　新規登録
住　所　大阪府門真市大字門真１００６番地
氏　名　松下電器産業株式会社